In [1]:

```
from typing import List, Optional
from tqdm import tqdm

import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt

import seaborn as sns
import scipy.stats as st
from scipy.stats import probplot, ks_2samp

from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import KFold, cross_val_score
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.utils.validation import check_is_fitted
import missingno as msno
import xgboost as xgb
%matplotlib inline
```

Базовый анализ данных

In [3]:

```
train = pd.read_csv("data2/train.csv")
test = pd.read_csv("data2/test.csv")

print("Размер train: rows: {}, cols: {}".format(*train.shape))
print("Размер test: rows: {}, cols: {}".format(*test.shape))
```

```
Размер train: rows: 200000, cols: 202
Размер test: rows: 200000, cols: 201
```

In [4]:

```
train.head()
```

Out[4]:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...

5 rows × 202 columns

In [5]:

```
test.head()
```

Out[5]:

	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	..
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	..
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	..
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	..
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	..
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	..

5 rows × 201 columns

In [6]:

```
train['target'].unique()
```

Out[6]:

```
array([0, 1], dtype=int64)
```

базовый анализ целевой переменной

Обучающая выборка содержит:

ID_code (строка, идентификатор записи); целевая переменная; 200 числовых переменных, названных от var_0 до var_199; Тестовая выборка содержит:

ID_code (строка, идентификатор записи); 200 числовых переменных, названных от var_0 до var_199;

In [10]:

```
tr = 0
ts = 0
for i in train['target']:
    if i == 0:
        ts += 1
    else:
        tr += 1
print("0: {}, 1: {}".format(ts, tr))
```

```
0: 179902, 1: 20098
```

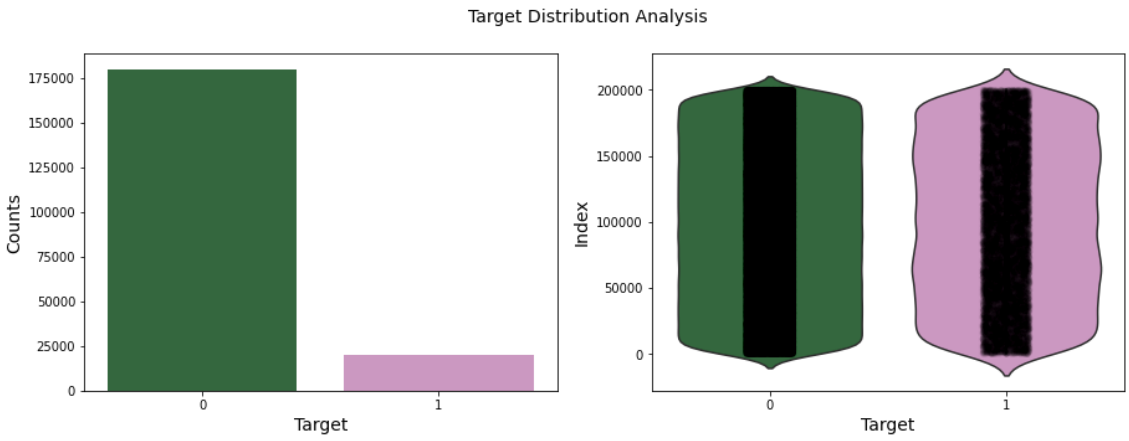
In [11]:

```
fig, axes = plt.subplots(1, 2, figsize=(15, 5))
plt.suptitle("Target Distribution Analysis", size=14)
print(f"Mean-target: {round(train.target.mean(), 4)}")

sns.countplot(train.target, ax=axes[0], palette="cubehelix")
sns.violinplot(train.target, train.index, ax=axes[1], palette="cubehelix")
sns.stripplot(train.target, train.index, jitter=True, ax=axes[1], color="black", alpha=0

axes[0].set_xlabel("Target", fontsize=14)
axes[0].set_ylabel("Counts", fontsize=14)
axes[1].set_xlabel("Target", fontsize=14)
axes[1].set_ylabel("Index", fontsize=14)
```

```
Mean-target: 0.1005

c:\program files\python37\lib\site-packages\seaborn\_decorators.py:43: Fu
tureWarning: Pass the following variable as a keyword arg: x. From versio
n 0.12, the only valid positional argument will be `data`, and passing ot
her arguments without an explicit keyword will result in an error or misi
nterpretation.
  FutureWarning
c:\program files\python37\lib\site-packages\seaborn\_decorators.py:43: Fu
tureWarning: Pass the following variables as keyword args: x, y. From ver
sion 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or mi
sinterpretation.
  FutureWarning
c:\program files\python37\lib\site-packages\seaborn\_decorators.py:43: Fu
tureWarning: Pass the following variables as keyword args: x, y. From ver
sion 0.12, the only valid positional argument will be `data`, and passing
other arguments without an explicit keyword will result in an error or mi
sinterpretation.
  FutureWarning
```

Out[11]:

```
Text(0, 0.5, 'Index')
```



очень сильная разница значений целевой переменной

In [12]:

```
count = 0
for i in train['ID_code']:
    if i.split("_")[0] == "test":
        count += 1
count
```

Out[12]:

```
0
```

In [13]:

```
count = 0
for i in test['ID_code']:
    if i.split("_")[0] == "train":
        count += 1
count
```

Out[13]:

```
0
```

In [14]:

```
train = train.drop("ID_code", axis=1)
```

In [15]:

```
train.sample()
```

Out[15]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	...	va
74591	0	9.9111	-1.6	11.8202	5.0792	9.9508	7.95	4.7129	10.7488	-2.3838	...	-1

1 rows × 201 columns

In [16]:

```
test.head()
```

Out[16]:

	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	..
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	..
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	..
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	..
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	..
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	..

5 rows × 201 columns

In [17]:

```
test = test.drop("ID_code", axis=1)
```

In [18]:

```
train.describe()
```

Out[18]:

	target	var_0	var_1	var_2	var_3	
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.(
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.(
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.(
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.(
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.(
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.(
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.(

8 rows × 201 columns

In [42]:

```
numerical_features = train.select_dtypes(include=[np.number])
print(f"count of numeric_features {numerical_features.shape[1]}")

numerical_features.columns
```

```
count of numeric_features 201
```

Out[42]:

```
Index(['target', 'var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'v
ar_6',
       'var_7', 'var_8',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=201)
```

Все признаки имеют числовые переменные

Анализ распределения признаков

In [20]:

```
features = train.drop("target",1).columns
```

```
c:\program files\python37\lib\site-packages\ipykernel_launcher.py:1: Futu
reWarning: In a future version of pandas all arguments of DataFrame.drop
except for the argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

In [21]:

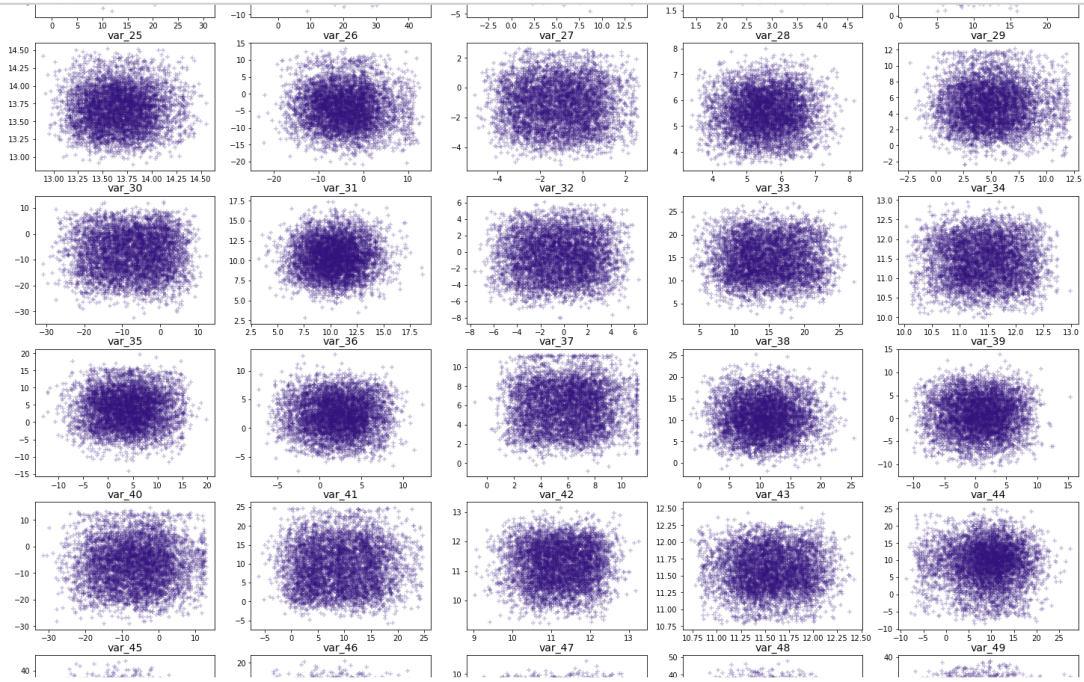
```
features
```

Out[21]:

```
Index(['var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'var_6', 'va
r_7',
       'var_8', 'var_9',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=200)
```

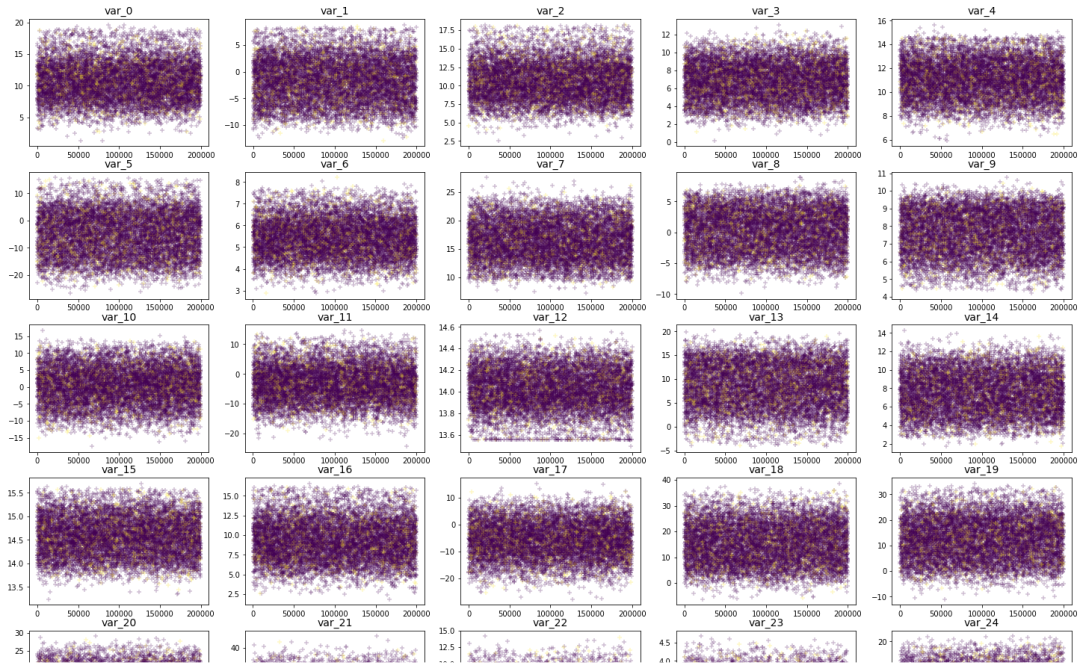
In [26]:

```
fig, axes = plt.subplots(40, 5, figsize=(25, 150))

for num, feature in enumerate(features, start=1):
    plt.subplot(40, 5, num)
    plt.title(f"{feature}", size=14)
    plt.scatter(
        train[feature].sample(5000),
        test[feature].sample(5000),
        color="#33137d",
        alpha=0.25,
        marker="+")
```



In [27]:

```
fig, axes = plt.subplots(40, 5, figsize=(25, 150))

for num, feature in enumerate(features, start=1):
    plt.subplot(40, 5, num)
    plt.title(f"{feature}", size=14)
    plt.scatter(
        train[feature].sample(10000).index,
        train[feature].sample(10000),
        c=train["target"].sample(10000),
        cmap="viridis",
        alpha=0.25,
        marker="+")
```



Анализ числовых переменных

Дескретные признаки

In [43]:

```
discrete_feature = [
    feature for feature in numerical_features
    if len(train[feature].unique())< 25
]

print(f"Discrete Variables Count: {len(discrete_feature)}")
```

```
Discrete Variables Count: 1
```

In [44]:

```
train[discrete_feature].head()
```

Out[44]:

	target
0	0
1	0
2	0
3	0
4	0

Непрерывные признаки

In [45]:

```
continuous_feature = [
    feature for feature in numerical_features
    if feature not in discrete_feature
]
print(f"Continuous Feature Count {len(continuous_feature)}")
```

```
Continuous Feature Count 200
```

In [46]:

```
train[continuous_feature].head()
```

Out[46]:

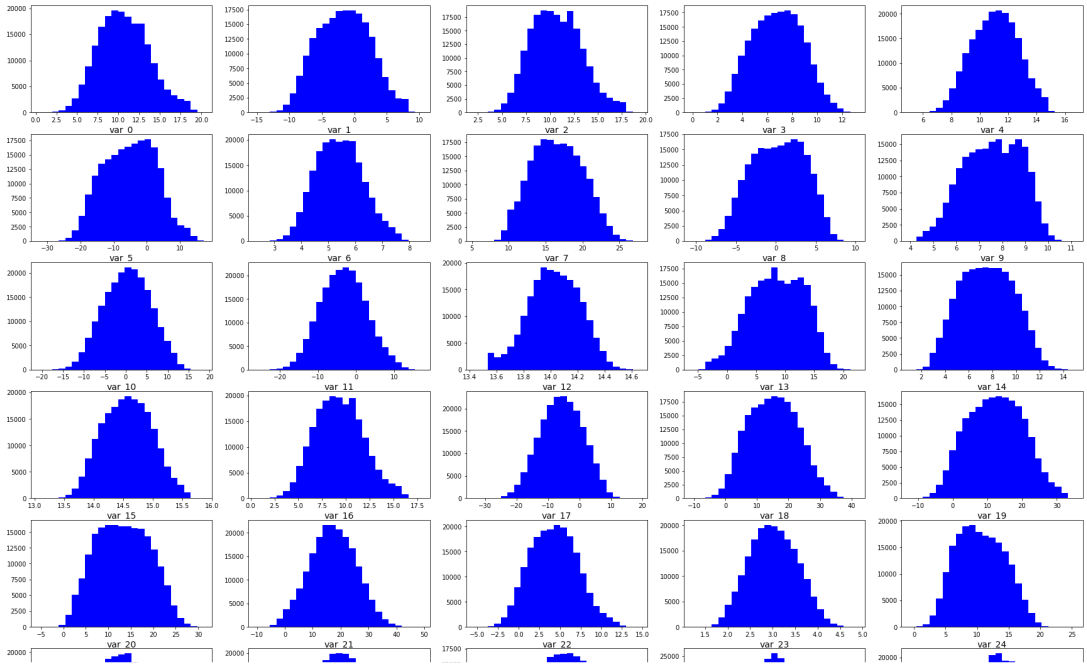
	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	...
0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	-4.9200	5.7470	...
1	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	3.1468	8.0851	...
2	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193	5.9525	...
3	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.8609	8.2450	...
4	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.2654	7.6784	...

5 rows × 200 columns

Гистрограммы распределения

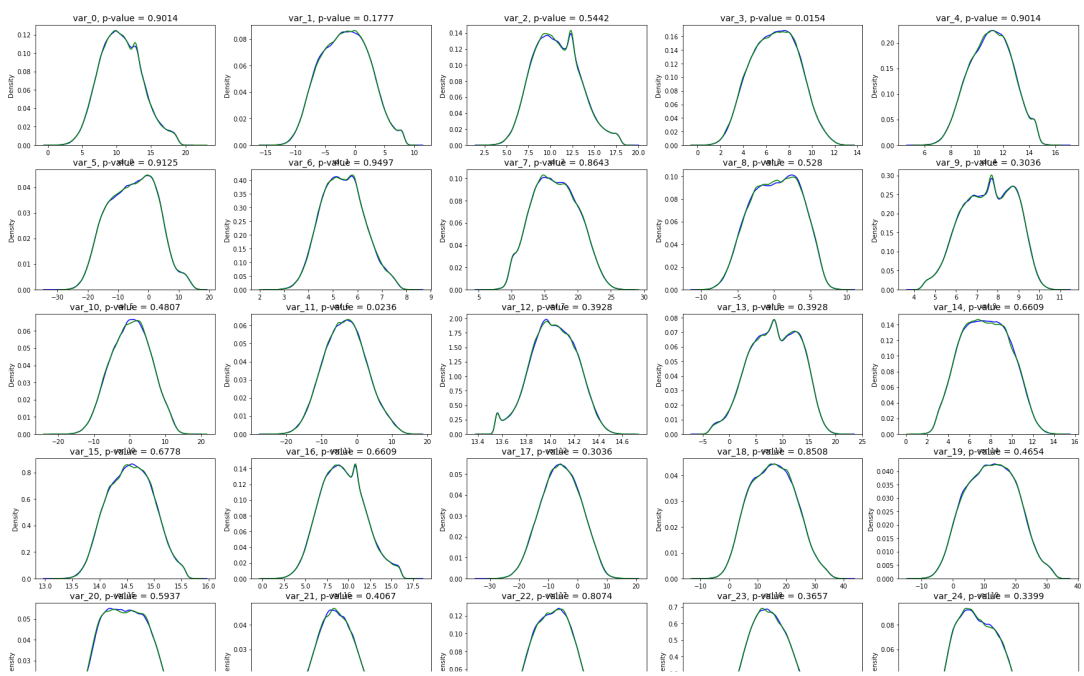
In [19]:

```
fig, axes = plt.subplots(40, 5,figsize=(30, 150))

for num, feature in enumerate(continuous_feature):
    data = train[feature].copy()
#     print(axes)
    axes[num//5, num%5].hist(data, bins=25, color="blue")
    axes[num//5, num%5].set_xlabel(feature, fontsize=14)
```



In [47]:

```
fig, axes = plt.subplots(40, 5, figsize=(30, 170))

for num, feature in enumerate(continuous_feature):
    try:
        statistic, pvalue = ks_2samp(train[feature].sample(5000), test[feature].sample(5
        train_data, test_data = train[feature].copy(), test[feature].copy()
        sns.kdeplot(train_data, ax=axes[num//5, num%5], color="blue", label="train")
        sns.kdeplot(test_data, ax=axes[num//5, num%5], color="green", label="test")
    except RuntimeError:
        pass
    except KeyError:
        train_data = train[feature].copy()
        sns.kdeplot(train_data, ax=axes[num//5, num%5], color="blue", label="train")
    axes[num//5, num%5].set_title(f"{feature}, p-value = {round(pvalue, 4)}", fontsize=1
```



In [49]:

```
columns = train.drop('target', axis= 1).columns
columns
```

Out[49]:

```
Index(['var_0', 'var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'var_6', 'va
r_7',
       'var_8', 'var_9',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=200)
```

In [50]:

```
train[columns[0:10]].shape[1]
```

Out[50]:

```
10
```

Связь между признаками

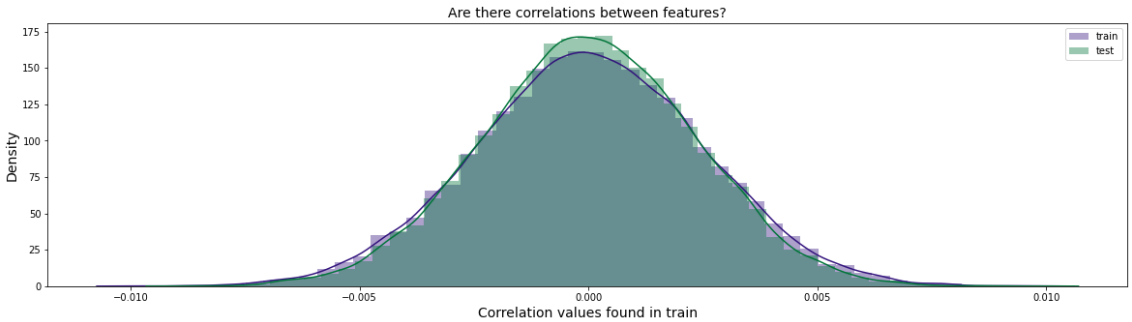
In [51]:

```
train_correlations = train.drop(["target"], axis=1).corr()
train_correlations = train_correlations.values.flatten()
train_correlations = train_correlations[train_correlations != 1]

test_correlations = test.corr()
test_correlations = test_correlations.values.flatten()
test_correlations = test_correlations[test_correlations != 1]

plt.figure(figsize=(20,5))
sns.distplot(train_correlations, color="#33137d", label="train")
sns.distplot(test_correlations, color="#007539", label="test")
plt.title("Are there correlations between features?", size=14)
plt.xlabel("Correlation values found in train", size=14)
plt.ylabel("Density", size=14)
plt.legend()
```

```
c:\program files\python37\lib\site-packages\seaborn\distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a fi
gure-level function with similar flexibility) or `histplot` (an axes-leve
l function for histograms).
  warnings.warn(msg, FutureWarning)
c:\program files\python37\lib\site-packages\seaborn\distributions.py:261
9: FutureWarning: `distplot` is a deprecated function and will be removed
in a future version. Please adapt your code to use either `displot` (a fi
gure-level function with similar flexibility) or `histplot` (an axes-leve
l function for histograms).
  warnings.warn(msg, FutureWarning)
```

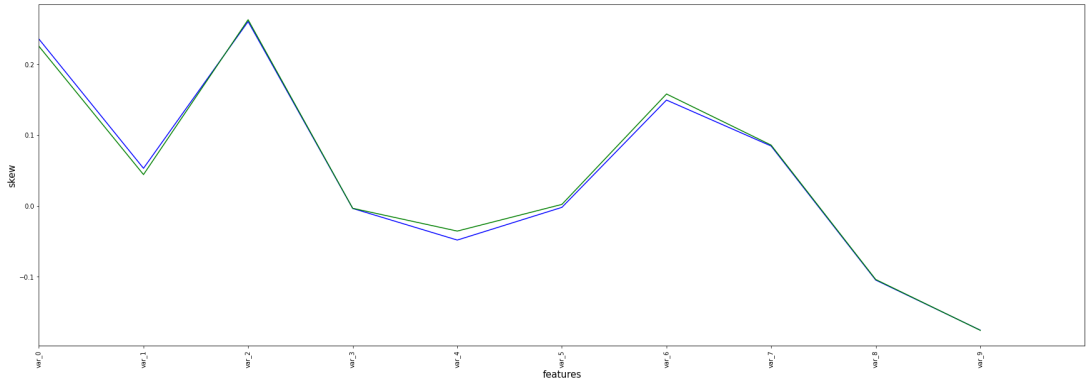
Out[51]:

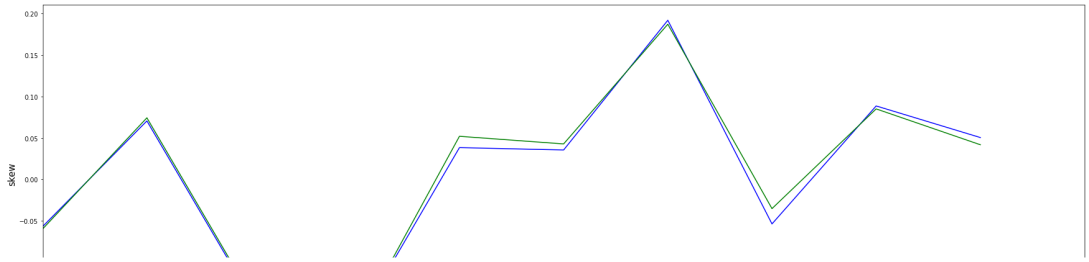
```
<matplotlib.legend.Legend at 0x1becff1db08>
```



In [53]:

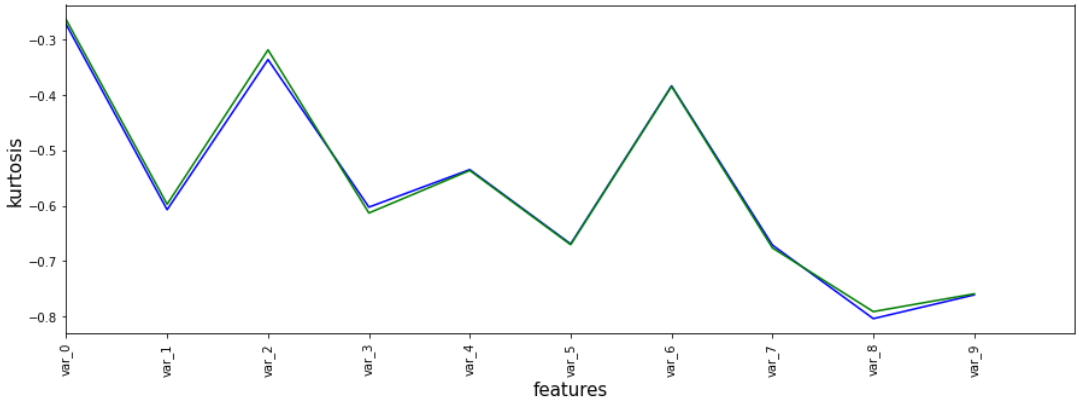
```
count_max = 10
count_min = 0
for i in range(len(columns)):
    if i % 10 == 0:
        col = columns[count_min:count_max]
        fig = plt.figure(figsize=(30, 10))
        train_stats, test_stats = train[col].skew(), test[col].skew()
        plt.plot(train_stats, color="blue", label="train")
        plt.plot(test_stats, color="green", label="test")
        plt.xticks(range(train_stats.shape[0]), train_stats.index, rotation=90)
        plt.xlabel("features", size=15)
        plt.xlim(0, len(train_stats))
        plt.ylabel("skew", size=15)
        plt.show()
        count_min +=10
        count_max +=10
```

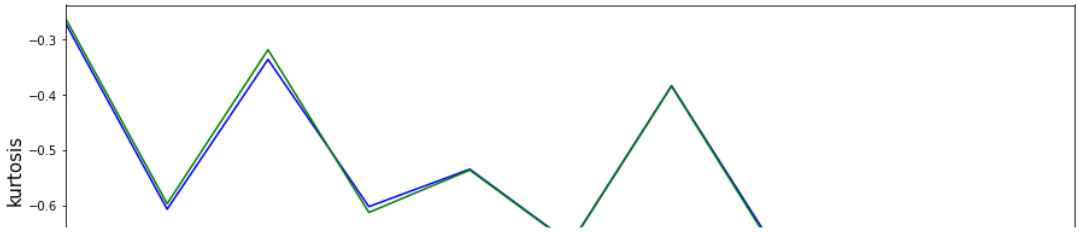




In [25]:

```
count_max = 10
count_min = 0
for i in range(len(columns)):
    if i % 10 == 0:
        col = columns[count_min:count_max]
        fig = plt.figure(figsize=(15, 5))
        train_stats, test_stats = train[col].kurtosis(), test[col].kurtosis()
        plt.plot(train_stats, color="blue", label="train")
        plt.plot(test_stats, color="green", label="test")
        plt.xticks(range(train_stats.shape[0]), train_stats.index, rotation=90)
        plt.xlabel("features", size=15)
        plt.ylabel("kurtosis", size=15)
        plt.xlim(0, len(train_stats))
        plt.show()
```





Type *Markdown* and LaTeX: α^2

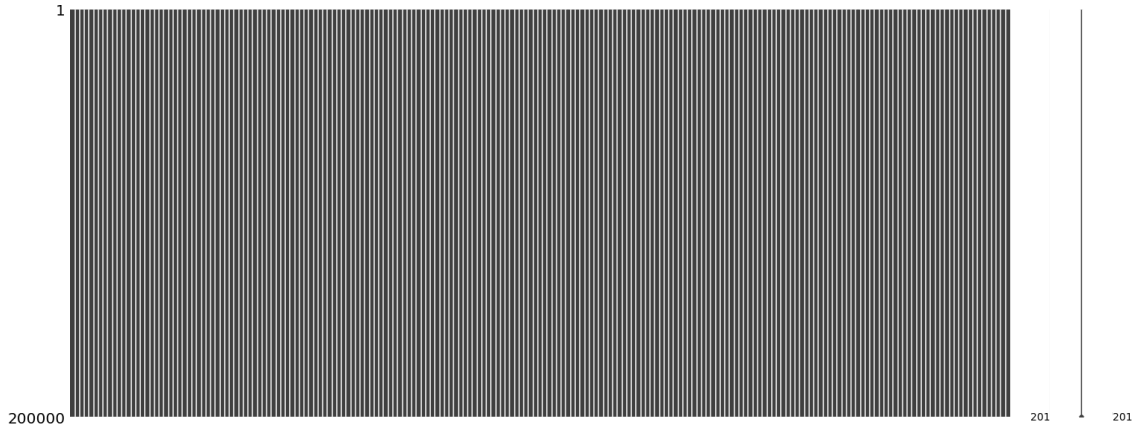
Анализ пропусков

In [26]:

```
msno.matrix(train)
```

Out[26]:

<AxesSubplot:>

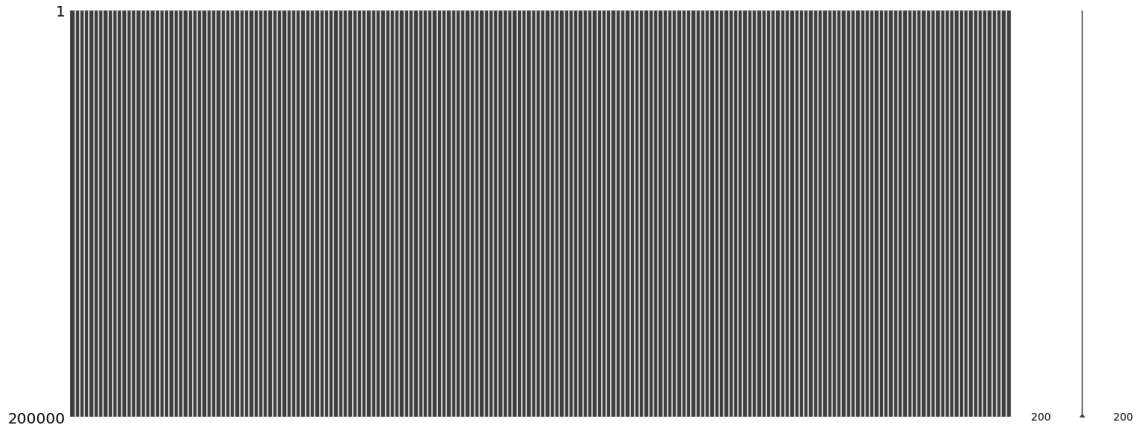


In [27]:

```
msno.matrix(test)
```

Out[27]:

<AxesSubplot:>



In [28]:

```
for n, i in enumerate(train.isnull().sum(axis= 1)):
    if i == 1:
        print(n, i)
```

Вывод по Базовому анализу данных

1) не равное распределение целевой переменной, "0" явно на много больше
2) в train нету тестовых данных, в test обучающих
3) все данные имеют числовые непрерывные признаки с нормальным распределением

4) данные на train и test индетичны, что очень хорошо

5) пропуски в данных отсутствуют, возможно как-то обработано

Корреляция между числовыми признаками

In [29]:

```
correlation = numerical_features.corr()
corr_with_target = correlation["target"].sort_values(ascending = False)
# corr_with_target
```

In [30]:

```
corr_with_target
```

Out[30]:

```
target     1.000000
var_6      0.066731
var_110    0.064275
var_53     0.063399
var_26     0.062422
             ...
var_76    -0.061917
var_146   -0.063644
var_12    -0.069489
var_139   -0.074080
var_81    -0.080917
Name: target, Length: 201, dtype: float64
```

In [31]:

```
for i, j in corr_with_target.items():
    print("{}:  {}".format(i,j))
```

```
target:  1.0
var_6:  0.0667308456127416
var_110:  0.06427529889100038
var_53:  0.0633986091422879
var_26:  0.06242219288600806
var_22:  0.0605584247221045
var_99:  0.05836701604941712
var_190:  0.05597341401301603
var_2:  0.05587034784189922
var_133:  0.05454762091998602
var_0:  0.052389591771425284
var_1:  0.05034262883427472
var_179:  0.05000176660313885
var_40:  0.04953025614409903
var_184:  0.0483154064438715
var_78:  0.04824466223983144
var_170:  0.04797278074621425
var_191:  0.04711367686893772
var_94:  0.046295765121515085
```

Correlation Heat Map

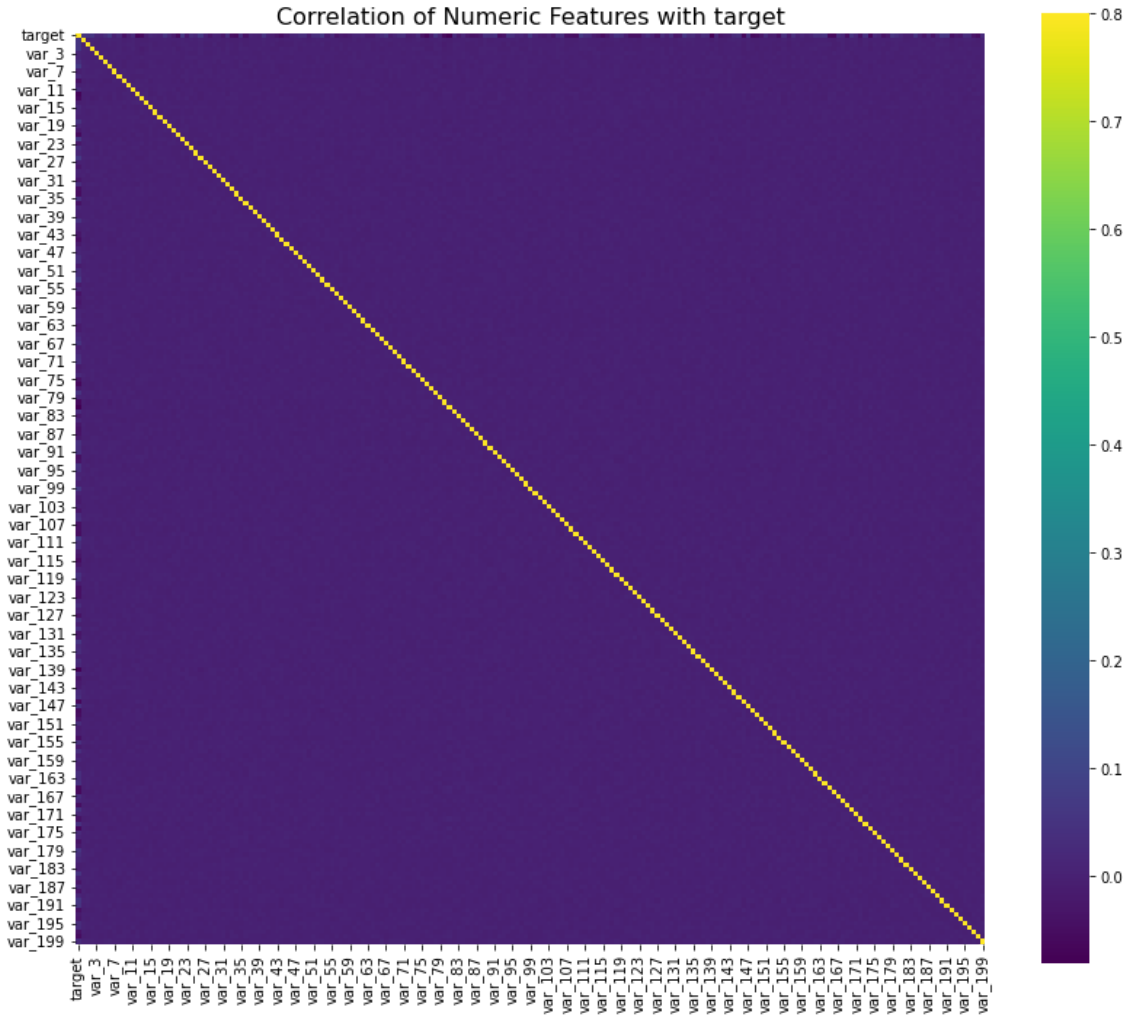
Тепловая карта - лучший способ быстро получить информацию о силе связи между переменными.

In [32]:

```
fig, axes = plt.subplots(figsize = (14,12))
plt.title("Correlation of Numeric Features with target", y=1, size=16)
sns.heatmap(correlation, square=True, vmax=0.8, cmap="viridis")
```

Out[32]:

```
<AxesSubplot:title={'center':'Correlation of Numeric Features with targe
t'}>
```



краткий вывод

Интересно! Коэффициент корреляции между всеми признаками близки к нулю, статистически не значимые результаты. Таким образом, все признаки не имеют линейной корреляции! Такая картина актуальна как для обучающей выборки.

Воспользуемся, например, случайным лесом, чтобы выбрать 15 основных признаков, которые связаны нелинейно с целевой переменной. Они могут служить отправной точкой для раскрытия природы данных и попыток понять данные. Кроме того, они могут дать некоторые идеи о том, как

создавать новые признаки. Будем использовать именно случайный лес, а не реализацию градиентого бустинга, потому что лес гораздо проще, он быстрее обучается и легче интерпретировать важность признаков случайного леса.

Нелинейная связь между признаками

Коэффициент корреляции позволяет установить линейную силу связи между признаками, но также признаки могут быть связаны нелинейно, что сложно определяется коэффициентом корреляции: может быть ситуация, что признаки очень сильно связаны между собой, но коэффициент корреляции равен 0.
Воспользуюсь случайным лесом, чтобы выбрать 15 наиболее значимых признаков

In [54]:

```
loc_num = np.log(np.fabs(numerical_features.drop("target", axis =1))+1)
```

In [55]:

```
np.fabs(-6.7863)
```

Out[55]:

```
6.7863
```

In [56]:

```
loc_num.head()
```

Out[56]:

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8
0	2.295107	2.052366	2.557855	1.807141	2.522580	2.330531	1.811350	2.976886	1.778336
1	2.525777	1.638472	2.698592	1.854578	2.592430	2.084839	1.890216	2.864130	1.422337
2	2.262731	1.320609	2.571123	2.185242	2.449495	2.310920	2.072253	2.748264	1.778218
3	2.489927	1.147974	2.297794	2.103610	2.608937	1.042430	1.923197	2.767890	1.925839
4	2.382957	0.909629	2.630060	2.033070	2.586048	1.237968	1.937374	3.008224	1.983123

5 rows × 200 columns

In [57]:

```
%%time
parameters = {"max_depth": 6, "n_estimators": 25, "random_state": 27, "n_jobs": 2}

forest = RandomForestRegressor(**parameters)
forest.fit(numerical_features.drop("target", axis=1), train['target'])
```

```
Wall time: 3min 56s
```

Out[57]:

```
RandomForestRegressor(max_depth=6, n_estimators=25, n_jobs=2, random_stat
e=27)
```

In [58]:

```
%%time
parameters = {"max_depth": 6, "n_estimators": 25, "random_state": 27, "n_jobs": 2}

forest_log = RandomForestRegressor(**parameters)
forest_log.fit(loc_num, train['target'])
```

```
Wall time: 3min 45s
```

Out[58]:

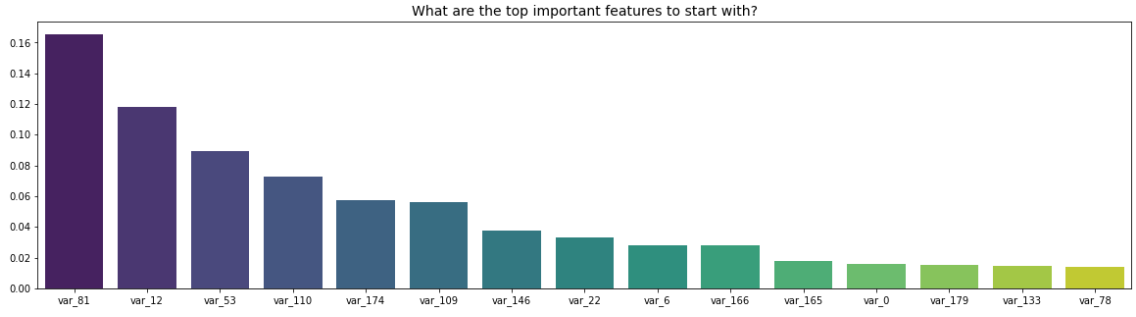
```
RandomForestRegressor(max_depth=6, n_estimators=25, n_jobs=2, random_stat
e=27)
```

In [61]:

```
# Log numerical
n_top = 15
importances = forest_log.feature_importances_
idx = np.argsort(importances)[::-1][0:n_top]
log_feature_names = loc_num.columns

plt.figure(figsize=(20, 5))
sns.barplot(x=log_feature_names[idx], y=importances[idx], palette="viridis")
plt.title("What are the top important features to start with?", size=14)
```

Out[61]:

```
Text(0.5, 1.0, 'What are the top important features to start with?')
```

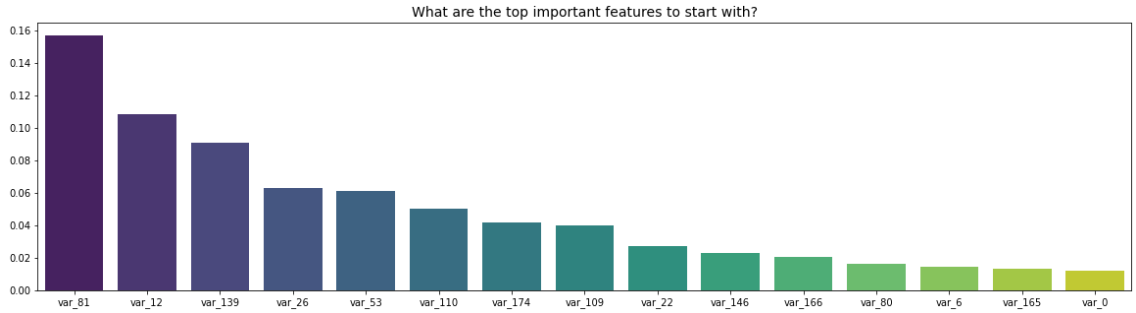


In [60]:

```
# numerical original
n_top = 15
importances = forest.feature_importances_
idx = np.argsort(importances)[::-1][0:n_top]
feature_names = numerical_features.drop("target", axis=1).columns

plt.figure(figsize=(20, 5))
sns.barplot(x=feature_names[idx], y=importances[idx], palette="viridis")
plt.title("What are the top important features to start with?", size=14)
```

Out[60]:

```
Text(0.5, 1.0, 'What are the top important features to start with?')
```



In [65]:

```
fig, axes = plt.subplots(n_top, 2, figsize=(15, 5*n_top))

for n in range(n_top):
    sns.kdeplot(
        train.loc[train.target==0, log_feature_names[idx][n]],
        ax=axes[n, 0],
        color="#33137d",
        label="target=0",
    )
    sns.kdeplot(
        train.loc[train.target==1, log_feature_names[idx][n]],
        ax=axes[n, 0],
        color="#007539",
        label="target=1",
    )
    sns.kdeplot(
        train.loc[:, log_feature_names[idx][n]],
        ax=axes[n, 1],
        color="#33137d",
        label="train"
    )
    sns.kdeplot(
        test.loc[:, log_feature_names[idx][n]],
        ax=axes[n, 1],
        color="#007539",
        label="test"
    )

    axes[n, 0].set_title("Train {}".format(log_feature_names[idx][n]))
    axes[n, 1].set_title("Test {}".format(log_feature_names[idx][n]))
    axes[n, 0].set_xlabel("")
    plt.legend(loc="best")
```

```
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
```

In [67]:

```
fig, axes = plt.subplots(n_top, 2, figsize=(15, 5*n_top))

for n in range(n_top):
    sns.kdeplot(
        train.loc[train.target==0, feature_names[idx][n]],
        ax=axes[n, 0],
        color="#33137d",
        label="target=0",
    )
    sns.kdeplot(
        train.loc[train.target==1, feature_names[idx][n]],
        ax=axes[n, 0],
        color="#007539",
        label="target=1",
    )
    sns.kdeplot(
        train.loc[:, feature_names[idx][n]],
        ax=axes[n, 1],
        color="#33137d",
        label="train"
    )
    sns.kdeplot(
        test.loc[:, feature_names[idx][n]],
        ax=axes[n, 1],
        color="#007539",
        label="test"
    )

    axes[n, 0].set_title("Train {}".format(feature_names[idx][n]))
    axes[n, 1].set_title("Test {}".format(feature_names[idx][n]))
    axes[n, 0].set_xlabel("")
    plt.legend(loc="best")
```

```
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
e label start with an underscore are ignored when legend() is called w
ith no argument.
No artists with labels found to put in legend.  Note that artists whos
```

Краткий вывод

Нелинейная связь сильнее чем линейная

У логарифма признаков, связь немного сильнее

Интересно, что на распределение признака при разном значении целевой переменной, накапливается некоторые пики, особенно для переменных var_81, var_12, var_53, var_109, var_6, var_0.