Задание

Для заданного набора данных (по Вашему варианту) постройте модели классификации или регрессии (в зависимости от конкретной задачи, рассматриваемой в наборе данных). Для построения моделей используйте методы 1 и 2 (по варианту для Вашей группы). Оцените качество моделей на основе подходящих метрик качества (не менее двух метрик). Какие метрики качества Вы использовали и почему? Какие выводы Вы можете сделать о качестве построенных моделей? Для построения моделей необходимо выполнить требуемую предобработку данных: заполнение пропусков, кодирование категориальных признаков, и т.д.

Метод 1: Линейная/логистическая регрессия

Метод 2: Градиентный бустинг

Набор данных: https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction

Данные были получены из Тайваньского экономического журнала за период с 1999 по 2009 год. Банкротство компании было определено на основании правил ведения бизнеса Тайваньской фондовой биржи.

Столбцы:

Y - Банкрот?: Ярлык класса

X1 - ROA(C) до вычета процентов и амортизации до вычета процентов: Рентабельность общих активов(C)

X2 - ROA(A) до уплаты процентов и % после налогообложения: Рентабельность общих активов(A)

X3 - ROA (B) до вычета процентов и амортизации после налогообложения: Рентабельность всех активов(B)

X4 – Валовая прибыль от операционной деятельности: Валовая прибыль/чистая выручка от продаж

X5 – Валовая прибыль от реализованных продаж: Реализованная валовая прибыль/чистая выручка от продаж

X6 – Норма операционной прибыли: Операционная прибыль/ Чистая выручка

X7 – Чистая процентная ставка до налогообложения: Доход/чистая выручка

X8 – Чистая процентная ставка после уплаты налогов: Чистая прибыль/Чистая выручка

X9 – Непромышленные доходы и расходы/выручка: Коэффициент чистой внереализационной прибыли

X10 — Непрерывная процентная ставка (после налогообложения): Чистая прибыль — исключая прибыль или убыток от выбытия / Чистая выручка

X11 — Ставка операционных расходов: Операционные расходы / Чистая выручка

X12 — Ставка расходов на исследования и разработки: (Расходы на исследования и разработки) / Чистая выручка

X13 - Скорость движения денежных средств: Денежный поток от операционных/текущих обязательств

X14 - Процентная ставка по процентным долгам: Процентные долги/капитал

X15 - Ставка налога (A): Эффективная налоговая ставка

X16 - Чистая стоимость одной акции (B): Балансовая стоимость На акцию(B)

X17 - Чистая стоимость на акцию (A): Балансовая стоимость на акцию(A)

X18 - Чистая стоимость на акцию (C): Балансовая стоимость на акцию(C)

X19 - Постоянная прибыль на акцию за последние четыре сезона: прибыль на акцию -Чистый доход

X20 - Денежный поток на акцию

X21 — Доход на акцию (в юанях): Продажи на акцию

X22 — Операционная прибыль на акцию (в юанях): Операционный доход на акцию

X23 — Чистая прибыль на акцию до налогообложения (в юанях): Доход на акцию до налогообложения X24 — Валовой реализованный доход Темп роста прибыли

X25 - Темп роста операционной прибыли: Рост операционной прибыли

X26 - Темп роста чистой прибыли после уплаты налогов: Рост чистой прибыли

X27 - Темп роста обычной чистой прибыли: Продолжающийся рост операционной прибыли после налогообложения

X28 - Темп постоянного роста чистой прибыли: Чистая прибыль - Исключая рост прибыли или убытков от выбытия

X29 - Темп роста общих активов: Рост совокупных активов

X30 - Темп роста чистой стоимости: Рост совокупных активов X31 - Коэффициент темпов роста общей доходности активов: Прибыль к совокупным приростам активов

X32 - Реинвестирование денежных средств, %: Коэффициент реинвестирования денежных средств

X33 - Коэффициент текущей ликвидности

X34 - Коэффициент быстрой ликвидности: тест на кислотность

X35 - Коэффициент процентных расходов: процентные расходы/общий доход

X35 - Коэффициент процентных расходов: процентные расходы/общий доход

X36 - Общий долг/Общий собственный капитал: Коэффициент общих обязательств/капитала

X37 - Коэффициент долга, % : Обязательства/Общие активы

X38 - Чистая стоимость/Активы: Собственный капитал/Общие активы

X39 - Коэффициент пригодности долгосрочного фонда (A): (Долгосрочные обязательства + Собственный капитал)/Основные активы

X40 - Зависимость от заимствования: Стоимость процентного долга

X41 – Условные обязательства/Собственный капитал: Условные обязательства/Капитал

X42 – Операционная прибыль/Оплаченный капитал: Операционный доход/Капитал

X43 – Чистая прибыль до налогообложения/Оплаченный капитал: Доход/капитал до налогообложения

X44 — Товарно-материальные запасы и дебиторская задолженность/Чистая стоимость: (Запасы+Дебиторская задолженность)/Собственный капитал

X45 — Общий оборот активов

X46 — Оборачиваемость дебиторской задолженности

X47 — Среднее количество дней сбора: Дни непогашенной дебиторской задолженности

X48 — Коэффициент оборачиваемости запасов (в разах)

X49 — Оборачиваемость основных средств Частота

X50 - Коэффициент оборота чистой стоимости (раз): Оборачиваемость собственного капитала

X51 - Выручка на человека: Продажи на одного работника

X52 - Операционная прибыль на человека: Операционный доход на одного работника

X53 - Коэффициент распределения на человека: Основные средства на одного работника

X54 - Общий оборотный капитал Активы

X55 - Быстрые активы/Итого активы

X56 - Оборотные активы/Итого активы

X57 - Денежные средства/Итого активы

X58 - Оборотные активы/текущие обязательства

X59 - Денежные средства/текущие обязательства

X60 - Текущие обязательства по активам

X61 - Оборотные средства по обязательствам

X62 - Запасы/оборотные средства

X63 - Запасы/текущие обязательства

X64 - Текущие обязательства/обязательства

X65 - Оборотные средства/капитал

X66 - Текущие обязательства/капитал

X67 - Долгосрочные обязательства по текущим активам

X68 - Нераспределенная прибыль к общей сумме активов

X69 - Общая прибыль/общая сумма расходов

X70 - Общая сумма расходов/активов

X71 - Коэффициент оборачиваемости текущих активов: Текущие активы к продажам

X72 - Быстрая оборачиваемость активов Отношение оборотных средств к продажам

X73 - Коэффициент оборачиваемости оборотного капитала: оборотный капитал к продажам

X74 - Норма денежного оборота: Денежные средства к продажам

X75 - Денежные потоки к продажам

X76 - Основные средства к активам

X77 - Текущие обязательства к обязательствам

X78 - Текущие обязательства к капиталу

X79 - Капитал к долгосрочным обязательствам

X80 - Денежные потоки к общим активам

X81 - Денежный поток к обязательствам

X82 - Финансовый директор к активам

X83 - Денежный поток к собственному капиталу

X84 - Текущие обязательства к оборотным активам

X85 - Флаг пассивов-активов: 1, если общая сумма обязательств превышает общую сумму активов, 0 в противном случае

X86 - чистая прибыль к общей сумме активов

X87 - общая сумма активов к цене ВНП

X88 - Интервал без кредита

X89 - Валовая прибыль к продажам

X90 - Чистая прибыль к акционерному капиталу

X91 - Обязательства к капиталу

X92 — Степень финансового рычага (DFL)

X93 — Коэффициент покрытия процентов (процентные расходы к EBIT)

X94 — Флаг чистой прибыли: 1, если чистая прибыль отрицательная за последние два года, 0 в противном случае

X95 — отношение капитала к обязательствам

Ход работы:

Подключаем все необходимые библиотеки:

In [2]:
```
import pandas as pd
import seaborn as sb
import numpy as np
import matplotlib
import matplotlib_inline
import matplotlib.pyplot as plt
from IPython.display import Image
from io import StringIO
import graphviz
import pydotplus
from sklearn.model_selection import train_test_split
%matplotlib inline
sb.set(style="ticks")
from IPython.display import set_matplotlib_formats
#matplotlib_inline.backend_inline.set_matplotlib_formats("retina")
```

Подключаем датасет

In [3]:
```
data = pd.read_csv('data.csv',sep =",")
```

Размер набора данных

In [4]:
```
data.shape
```

Out[4]:(6819, 96)

Типы колонок

In [5]:
```
data.dtypes
```

Out[5]:
```
Bankrupt?                                                 int64
 ROA(C) before interest and depreciation before interest    float64
 ROA(A) before interest and % after tax                     float64
 ROA(B) before interest and depreciation after tax          float64
 Operating Gross Margin                                     float64
                                                            ...
 Liability to Equity                                        float64
 Degree of Financial Leverage (DFL)                         float64
 Interest Coverage Ratio (Interest expense to EBIT)         float64
 Net Income Flag                                            int64
 Equity to Liability                                        float64
Length: 96, dtype: object
```

Проверяем есть ли пропущенные значения

In [6]:
```
data.isnull().sum()
```

Out[6]:Bankrupt? 0
ROA(C) before interest and depreciation before interest 0
ROA(A) before interest and % after tax 0
ROA(B) before interest and depreciation after tax 0
Operating Gross Margin 0
..
Liability to Equity 0
Degree of Financial Leverage (DFL) 0
Interest Coverage Ratio (Interest expense to EBIT) 0
Net Income Flag 0
Equity to Liability 0
Length: 96, dtype: int64

В наборе нет пропусков, следовательно не нужно их обрабатывать.

Возьмем для анализа первые 2000 строк набора данных

In [7]:data_2t = data.head(2000)

Первые 5 строк датасета

In [8]:data_2t.head(5)

Out[8]:

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...

5 rows × 96 columns

In [9]:
```
from IPython.display import set_matplotlib_formats
#matplotlib.use('nbagg')
#print(matplotlib.get_backend())
#matplotlib_inline.backend_inline.set_matplotlib_formats("retina")
pd.set_option("display.width", 70)
```

Нет категориальных значений, значит ненужно кодировать категориальных признаков.

Масштабирование данных

In [10]:
```
from sklearn.preprocessing import MinMaxScaler
scl = MinMaxScaler ()
scl_data = scl.fit_transform(data_2t)
data_scalled = data_2t.copy()
data_scalled[data_scalled.columns] = scl_data
data_scalled
```

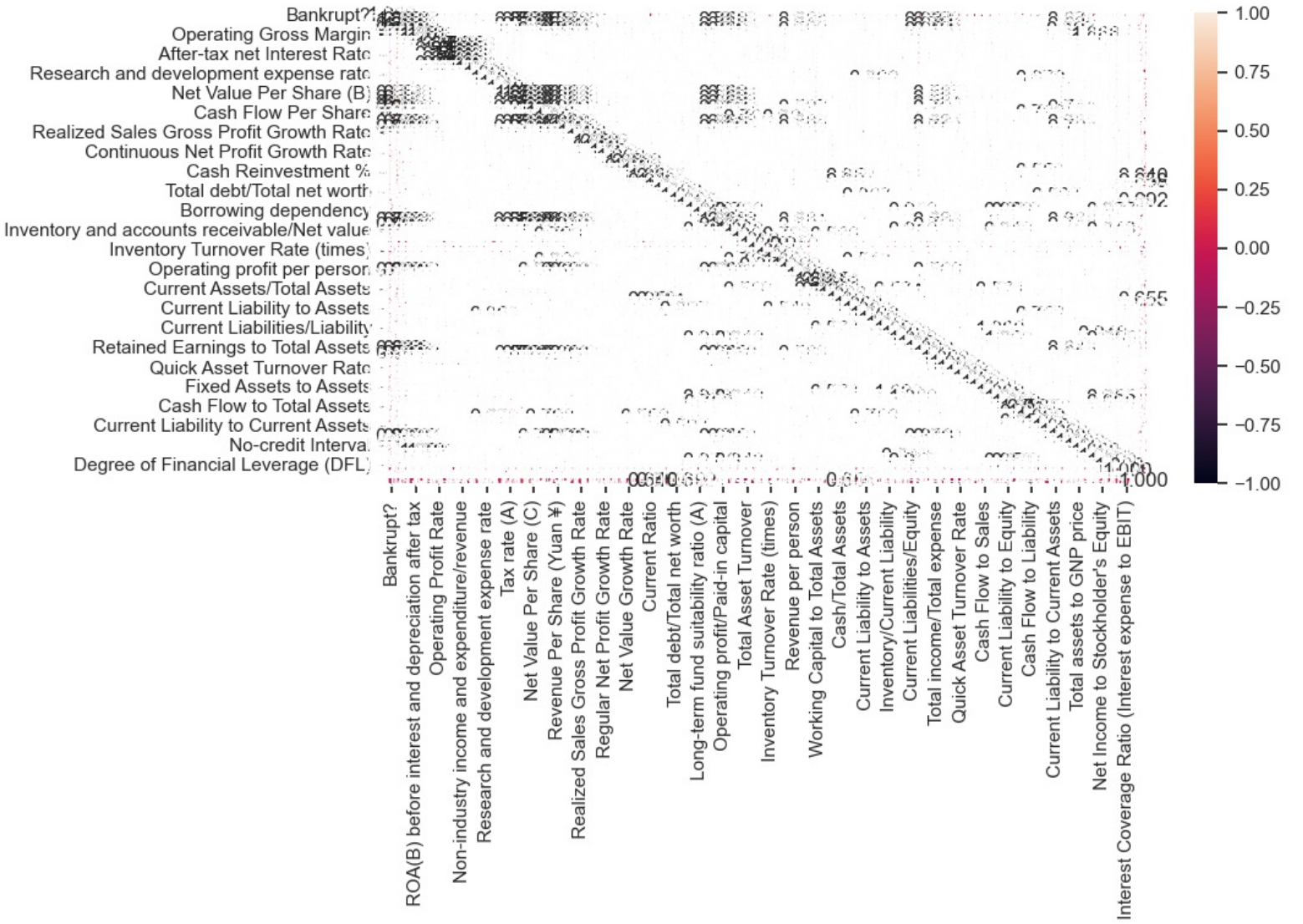
Out[10]:

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue
0	1.0	0.505452	0.564662	0.552325	0.874827	0.874827	0.981222	0.934741	0.955346	0.827681
1	1.0	0.633245	0.716109	0.703396	0.892077	0.892077	0.980329	0.945235	0.965279	0.851899
2	1.0	0.581117	0.663959	0.642909	0.874812	0.874642	0.976925	0.924444	0.946819	0.811414
3	1.0	0.545346	0.600421	0.623087	0.839617	0.839617	0.970871	0.936440	0.958505	0.846393
4	1.0	0.634242	0.716400	0.710975	0.869572	0.869572	0.981372	0.944934	0.965340	0.849732
...	...	...	...	...	...	...	...	...	...	...
1995	0.0	0.652128	0.661275	0.700554	0.878311	0.878311	0.977453	0.941628	0.961634	0.848188
1996	0.0	0.551995	0.606804	0.623160	0.869374	0.869317	0.981559	0.941316	0.962266	0.841556
1997	0.0	0.651662	0.713353	0.717097	0.882857	0.880817	0.982464	0.945449	0.965524	0.849275
1998	0.0	0.697207	0.753318	0.753170	0.884103	0.884967	0.982455	0.950381	0.968732	0.860062
1999	0.0	0.710306	0.747008	0.770223	0.879543	0.879543	0.982748	0.948276	0.967772	0.855039

2000 rows × 96 columns

In [11]:
```
ig, ax = plt.subplots(figsize=(10,5))
sb.heatmap(data_scalled.corr(method='pearson'),ax=ax, annot=True, fmt='.3f')
```

Out[11]:<Axes: >



In [12]:*#data_scalled.dtypes*

```
X = data_scalled.drop(columns=' ROA(C) before interest and depreciation before interest')
Y = data_scalled[' ROA(C) before interest and depreciation before interest']
```

In [13]:X.head()

Out[13]:

	Bankrupt?	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)	...
0	1.0	0.564662	0.552325	0.874827	0.874827	0.981222	0.934741	0.955346	0.827681	0.949485	...
1	1.0	0.716109	0.703396	0.892077	0.892077	0.980329	0.945235	0.965279	0.851899	0.960267	...
2	1.0	0.663959	0.642909	0.874812	0.874642	0.976925	0.924444	0.946819	0.811414	0.934983	...
3	1.0	0.600421	0.623087	0.839617	0.839617	0.970871	0.936440	0.958505	0.846393	0.954784	...
4	1.0	0.716400	0.710975	0.869572	0.869572	0.981372	0.944934	0.965340	0.849732	0.961179	...

5 rows × 95 columns

In [14]:Y.head()

Out[14]:0 0.505452
1 0.633245
2 0.581117
3 0.545346
4 0.634242
Name: ROA(C) before interest and depreciation before interest, dtype: float64

In [15]:X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state = 2022, test_size = 0.1)

In [16]:X_train.head()

Out[16]:

	Bankrupt?	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)
1964	0.0	0.666933	0.677015	0.867561	0.867901	0.980312	0.943662	0.964190	0.848497	0.960128
1510	0.0	0.762748	0.762717	0.882460	0.882460	0.982868	0.946562	0.966593	0.851121	0.962590
228	0.0	0.721622	0.734587	0.872263	0.872263	0.982580	0.945691	0.965832	0.849635	0.961916
1189	0.0	0.739900	0.741801	0.871427	0.871427	0.982837	0.945956	0.965977	0.849841	0.962024
1889	0.0	0.673243	0.668051	0.872872	0.872872	0.982109	0.944286	0.964645	0.847255	0.960642

5 rows × 95 columns

In [17]:X_test.head()

Out[17]:

	Bankrupt?	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)
1018	0.0	0.691231	0.740490	0.911240	0.907996	0.984657	0.944793	0.964245	0.844663	0.964132
1295	0.0	0.755639	0.755502	0.873580	0.873580	0.983494	0.947260	0.966832	0.851732	0.963089
643	0.0	0.744614	0.776126	0.895519	0.895519	0.984089	0.947007	0.966885	0.850324	0.963246
1842	0.0	0.722057	0.725404	0.877602	0.877602	0.982517	0.945252	0.965569	0.848767	0.961661
1669	0.0	0.762167	0.775179	0.916735	0.916735	0.986392	0.949301	0.973196	0.851996	0.969847

5 rows × 95 columns

In [18]:Y_train.head()

Out[18]:1964 0.609309
1510 0.696077
228 0.670545
1189 0.669947
1889 0.598670
Name: ROA(C) before interest and depreciation before interest, dtype: float64

In [19]:Y_test.head()

Out[19]:1018 0.679388
1295 0.692553
643 0.701862
1842 0.657646
1669 0.678590
Name: ROA(C) before interest and depreciation before interest, dtype: float64

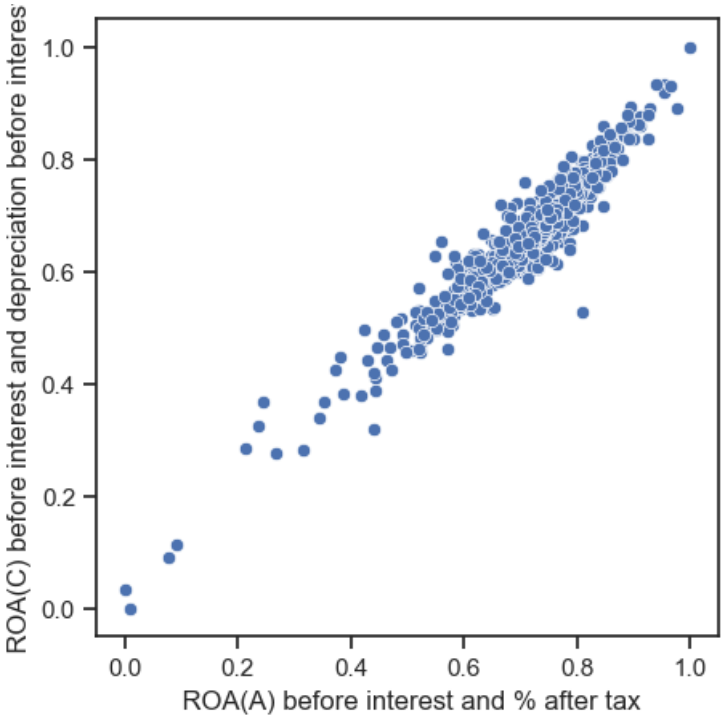
In [ ]:

Линейная регрессия

In [20]:**from** sklearn.linear_model **import** LinearRegression
from sklearn.datasets **import** make_blobs
from sklearn.metrics **import** mean_absolute_error, mean_squared_error

In [21]:fig, ax = plt.subplots(figsize=(5,5))
sb.scatterplot(ax=ax, x=X[' ROA(A) before interest and % after tax'], y=Y)

Out[21]:<Axes: xlabel=' ROA(A) before interest and % after tax', ylabel=' ROA(C) before interest and depreciation before interest'>



In [22]:reg1 = LinearRegression().fit(X, Y)

In [23]:Y_pred_1 = reg1.predict(X_test)

In [24]:mean_absolute_error(Y_test, Y_pred_1), mean_squared_error(Y_test, Y_pred_1)

Out[24]:(0.004226886505776256, 3.150647204460351e-05)

Градиентный бустинг

In [25]:**from** sklearn.ensemble **import** AdaBoostRegressor
from sklearn.tree **import** DecisionTreeClassifier, DecisionTreeRegressor, export_graphviz

In [26]:ab1 = AdaBoostRegressor(n_estimators=4, random_state=2022)
ab1.fit(X, Y)

Out[26]:

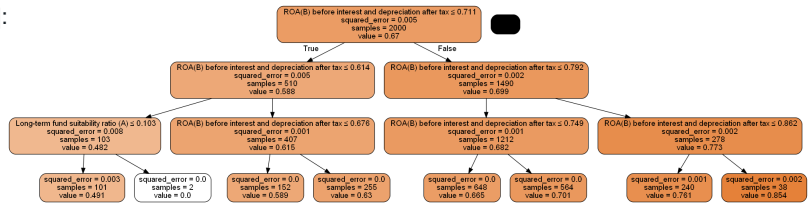
AdaBoostRegressor

AdaBoostRegressor(n_estimators=4, random_state=2022)

In [27]:*# Визуализация дерева*
def get_png_tree(tree_model_param, feature_names_param):
dot_data = StringIO()
export_graphviz(tree_model_param, out_file=dot_data, feature_names=feature_names_param,
filled=**True**, rounded=**True**, special_characters=**True**)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
return graph.create_png()

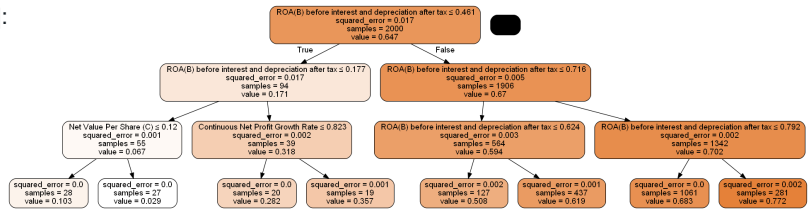
In [28]:Image(get_png_tree(ab1.estimators_[0], X.columns), width="500")

Out[28]:



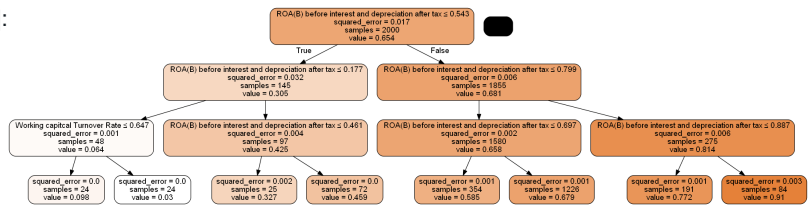
In [29]:Image(get_png_tree(ab1.estimators_[1], X.columns), width="500")

Out[29]:



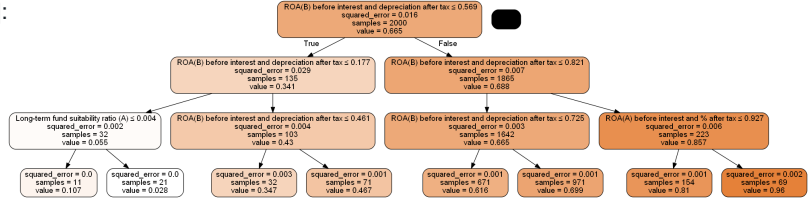
In [30]:Image(get_png_tree(ab1.estimators_[2], X.columns), width="500")

Out[30]:



In [31]:Image(get_png_tree(ab1.estimators_[3], X.columns), width="500")

Out[31]:



In [32]:
```
regressor = AdaBoostRegressor(n_estimators=4, random_state=2022)
regressor.fit(X_train, Y_train)
y_pred = regressor.predict(X_test)
```

In [33]:
```
print('Mean Absolute Error:', mean_absolute_error(Y_test, y_pred))
print('Mean Squared Error:', mean_squared_error(Y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(Y_test, y_pred)))
```

Mean Absolute Error: 0.012428410268481134
Mean Squared Error: 0.0002818729129094282
Root Mean Squared Error: 0.016789071234271067
Как видно, линейная регрессия показала более лучшие результаты, чем градиентный бустинг.