Министерство науки и высшего образования Российской Федерации

Федеральное Государственное Бюджетное Образовательное Учреждение

Высшего Образования

«Российский экономический университет имени Г. В. Плеханова»

Кафедра математических методов в экономике

Выпускная квалификационная работа

по программе профессиональной переподготовке

«Основы Data Science на языке Python»

на тему «Прогнозный анализ поведения клиентов банка»

Выполнил:

Суворов Александр Александрович

Преподаватель:

Профессор кафедры математических методов в экономике

Ph.D. in economics

Моисеев Никита Александрович

Москва

2022

Оглавление

1. Понимание бизнес-целей

Как показывает практика, наиболее эффективным способ анализа информации является методы и алгоритмы анализа данных. Анализ базы данных представляет актуальность и научно - практический интерес.

Целью выпускной квалификационной работы является исследование особенностей практического применения методов разработки модели анализа поведения клиентов банка на основании методов машинного обучения.

Объектом исследования является поведение клиентов банка.

Предметом исследования являются методы машинного обучения.

Практическая значимость работы заключается в разработке моделей на базе машинного обучения, позволяющих осуществлять своевременный прогноз ухода клиентов банка в отток.

Данная выпускная квалификационная работа будет рассмотрена на примере прогноза поведения клиентов.

1.1 Доступные ресурсы

Для успешной реализации проекта необходимы следующие категории специалистов: аналитик данных, специалист по базам данных, руководитель проекта.

Заказчик располагает всем необходимым оборудованием для проведения анализа данных.

1.2 Риски

1. Несоблюдение сроков проекта;
2. Риск неплатежеспособности заказчика исследования;
3. Риск нехватки и неполноты данных;
4. Риск несоответствия полученных результатов требованиям заказчика исследования;

1.3 Ограничения

Ограничение сроков: 6 месяцев (проект анализа данных)

Ставки по сотрудникам:

Аналитик данных – 1 ставка;

Специалист по базам данных – 0,5 ставки;

Руководитель проекта - 1 ставка.

1.4 Цели исследования данных

Задачи анализа данных, решаемые в рамках проекта:

1. Сбор данных, пре - процессинг (очистка, проверка на выбросы, дубликаты и т.д.).
2. Построение модели и интерпретация полученных результатов.
3. Результаты работы могут быть рекомендованы для бизнес – аналитиков и разработчиков программ, используемых для управленческих решений в банковской сфере.

1.5 Критерии успешности изучения данных

Метрики оценки точности и качества построенных моделей:

Основной метрикой качества задачи классификации будет являться площадь под ROC - кривой.

Границы значений метрик: площадь под ROC – кривой колеблется в диапазоне от 0,5 до 1.

2. Начальное изучение данных

2.1 Сбор данных

Внутренние данные: RowNumber (номер строки), CustomerId (номер клиента), Surname (Фамилия), CreditScore (кредитный рейтинг), Geography (география, страна проживания клиента), Gender (Пол), Age (Возраст), Tenure (срок пребывания в должности), Balance (Остаток средств), NumOfProducts (количество банковских продуктов, используемых клиентом), HasCrCard (наличие кредитной карты), IsActiveMember (является активным пользователем), EstimatedSalary (Расчетная зарплата), Exited (уход клиента из банка).

Внешние данные: отсутствуют;

Дополнительные данные: отсутствуют.

2.2 Описание данных

Объем данных (размер файла) – 268 Кбайт, 10000 строк, 14 колонок.

Это набор данных клиентов банка США, который характеризует клиентов данного банка и содержит информацию о том, покинул этот клиент банк или нет.

Источник данных: открытые данные с сайта https://www.kaggle.com/datasets/shantanudhakadd/bank-customer-churn-prediction

Типы, виды данных и схемы кодирования приведены в таблице 1:

Таблица 1. Типы, виды данных и схемы кодирования

№	*Наименование*	*Тип данных*	*Вид данных*	*Схема кодирования*
1	RowNumber	Целое число (integer)	Непрерывный	Отсутствует
2	CustomerId	Целое число (integer)	Непрерывный	Отсутствует
3	Surname	Строковый (object)		Отсутствует
4	CreditScore	Целое число (integer)	Дискретный	Отсутствует
5	Geography	Строковый (object)	Дискретный	Фиктивные переменные
6	Gender	Строковый (object)	Дискретный	Фиктивные переменные
7	Age	Целое число (integer)	Непрерывный	Отсутствует
8	Tenure	Целое число (integer)	Непрерывный	Отсутствует
9	Balance	Число с плавающей запятой (float)	Непрерывный	Отсутствует
10	NumOfProducts	Целое число (integer)	Дискретный	Отсутствует

№	*Наименование*	*Тип данных*	*Вид данных*	*Схема кодирования*
11	HasCrCard	Целое число (integer)	Дискретный	Отсутствует
12	IsActiveMember	Целое число (integer)	Дискретный	Отсутствует
13	EstimatedSalary	Число с плавающей запятой (float)	Непрерывный	Отсутствует
14	Exited	Целое число (integer)	Дискретный	Отсутствует

Формат данных – файл csv, разделитель “ , ”, название файла – “ Churn_Modelling”.

2.3 Исследование данных

После первичного анализа данных было принято решение удалить следующие столбцы:

1. RowNumber(номер строки)
2. CustomerId (номер клиента)
3. Surname (Фамилия)

Описательная статистика полученного набора данных представлена в таблице 2:

Таблица 2. Построение описательной статистики исходного набора данных

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000	10000	10000	10000	10000	10000	10000	10000	10000
mean	650,53	38,92	5,01	76 485,89	1,53	0,71	0,52	100 090,24	0,20
std	96,65	10,49	2,89	62 397,41	0,58	0,46	0,50	57 510,49	0,40
min	350	18	0	0	1	0	0	11,58	0
25%	584	32	3	0	1	0	0	51002,11	0
50%	652	37	5	97198,54	1	1	1	100193,915	0
75%	718	44	7	127644,24	2	1	1	149388,2475	0
max	850	92	10	250898,09	4	1	1	199992,48	1

Пропущенные значения отсутствуют.
Дубликаты (одинаковые строки) отсутствуют.
Столбцы с полностью одинаковыми значениями отсутствуют.
Выбросы отсутствуют.

Для наглядности построены диаграммы «Ящик с усами», которые представлены на рисунках 1 – 4.

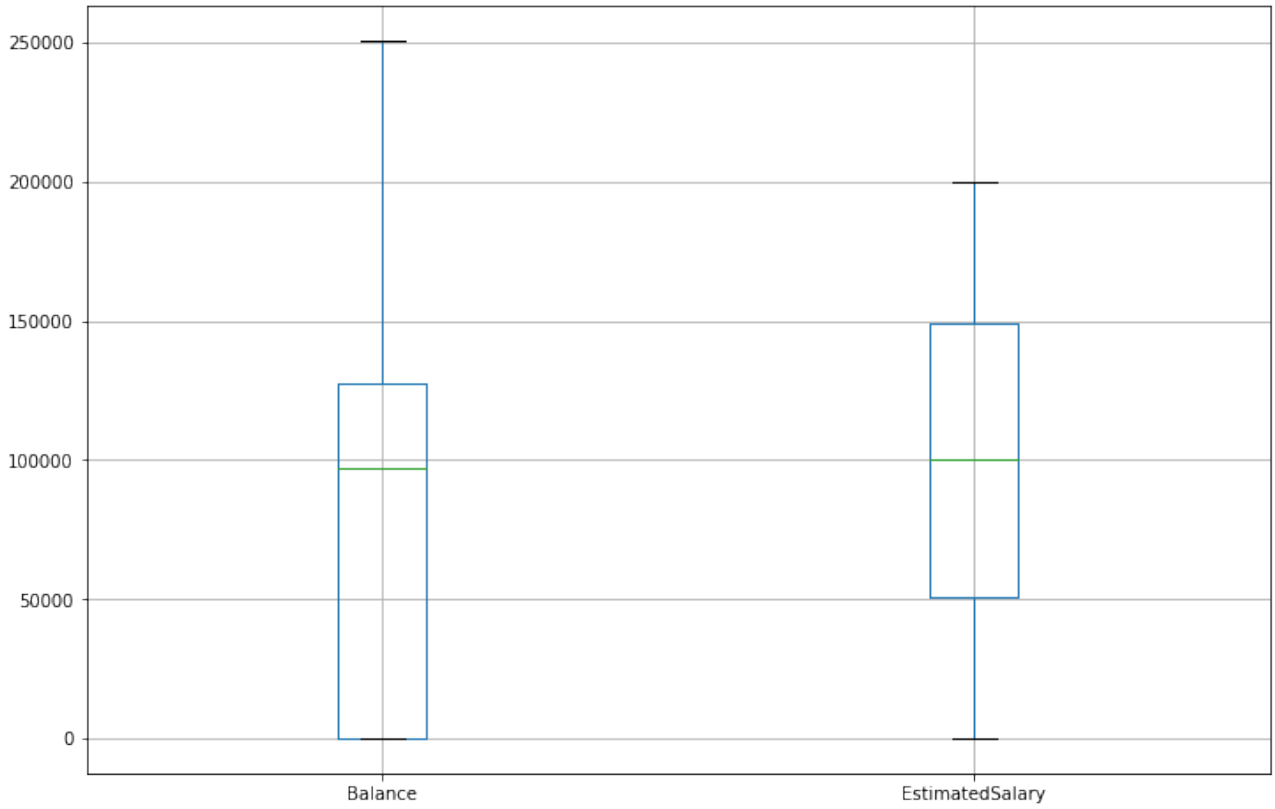

Рис. 1. Диаграмма «Ящик с усами» для столбцов Balance и EstimatedSalary

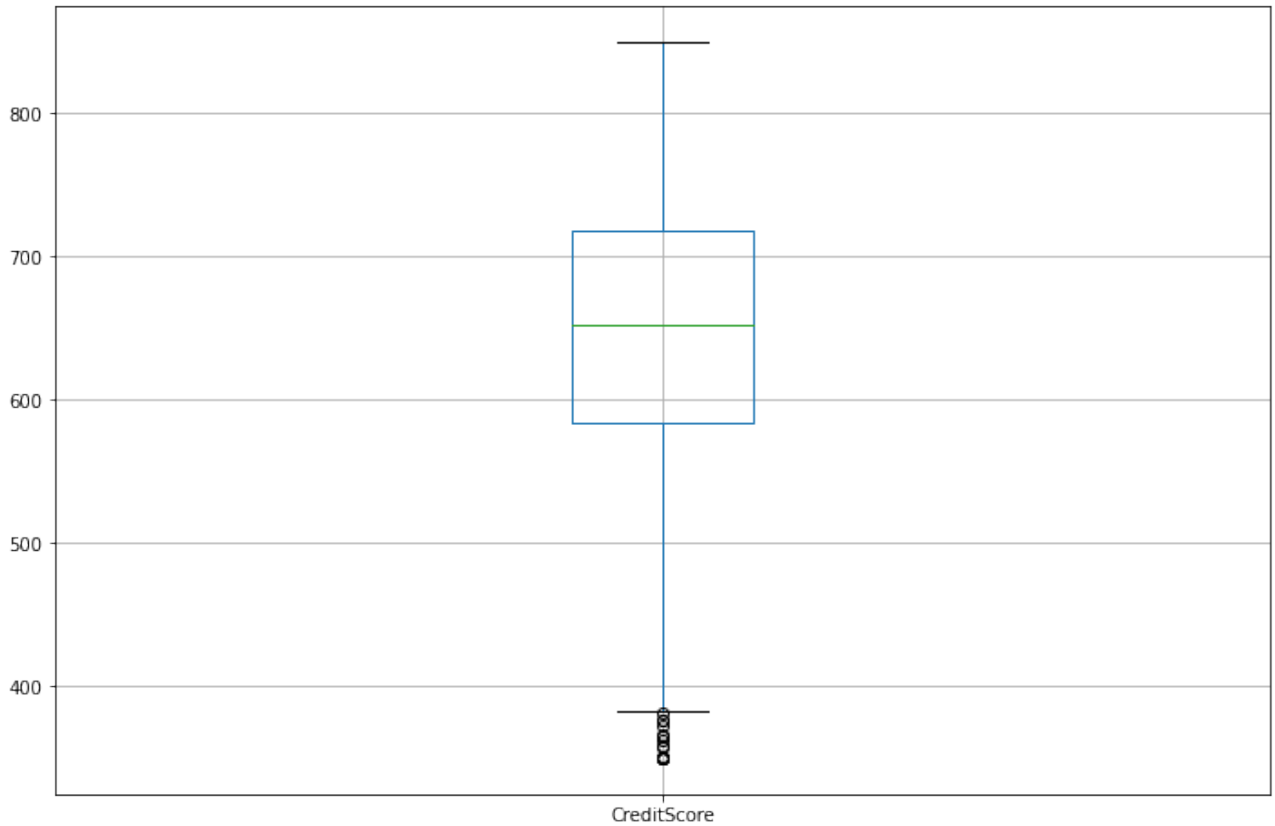

Рис. 2. Диаграмма «Ящик с усами» для столбца CreditScore

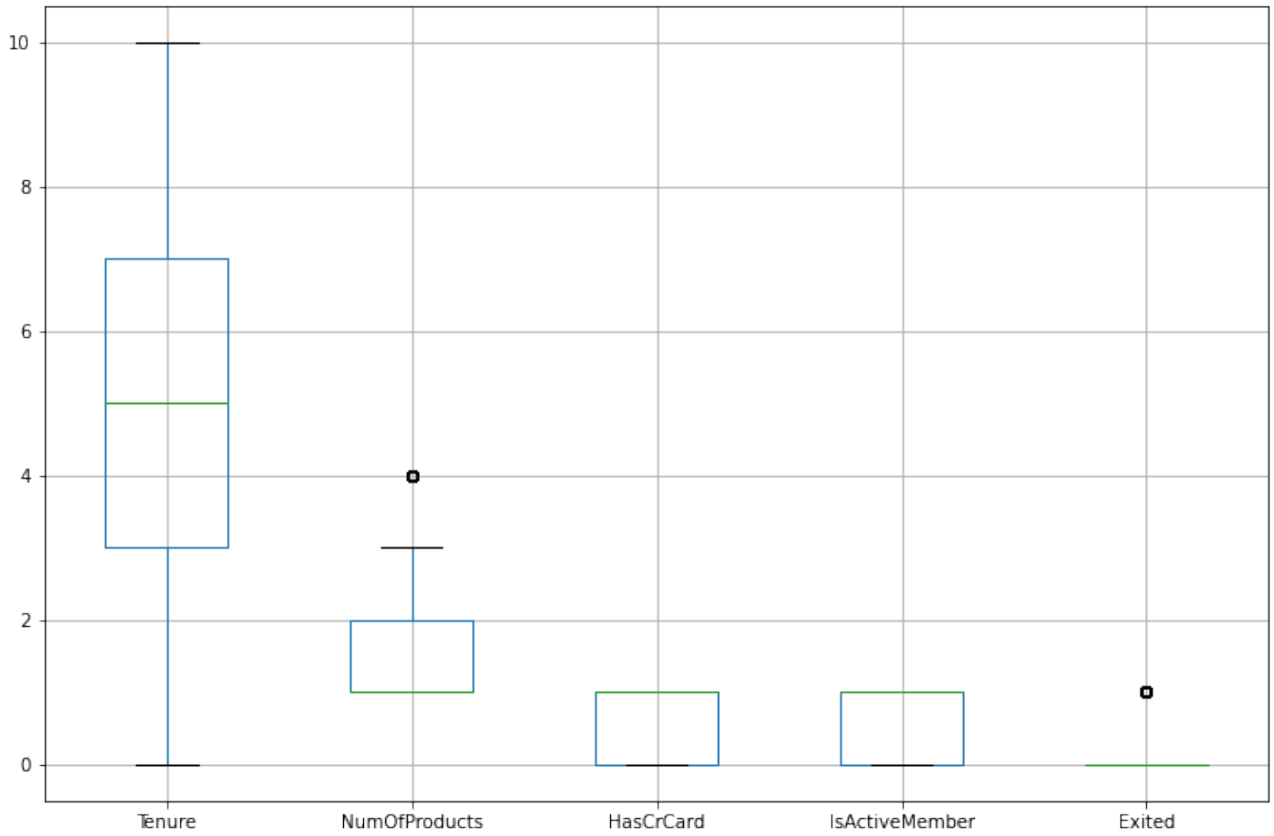

Рис. 3. Диаграмма «Ящик с усами» для столбцов Tenure NumOfProducts HasCrCard IsActiveMember Exited

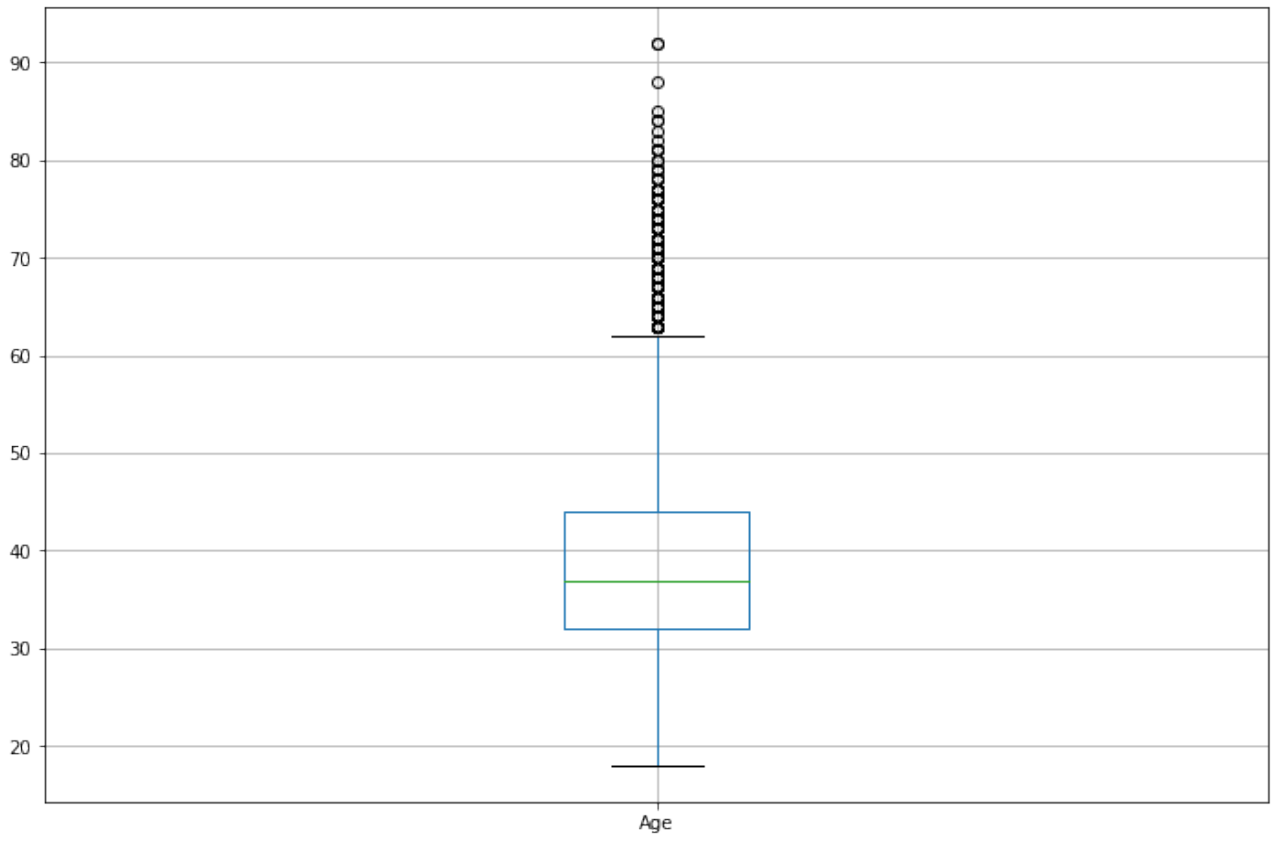

Рис. 4. Диаграмма «Ящик с усами» для столбца Age

Из диаграммы «Ящик с усами» для столбца Age видно, что в данных присутствует несколько выбросов. Однако максимальное значение выброса составило 92 года. Это говорит о допустимости подобных значений. Поэтому было принято решение данные выбросы оставить.

Производим преобразование категориальных переменных по методу фиктивных переменных: Geography (География, Страна проживания клиента), Gender (Пол).

Для оценки тесноты связи между рассматриваемыми показателями и выявления **мультиколлинеарности** между факторами модели построена матрица корреляции. Затем было принято решение удалить столбцы с корреляцией больше 0,5 ('Geography_France', 'Geography_Spain').

Итоговая корреляционная матрица представлена в Таблице 3:

Таблица 3. Матрица корреляции

	Exited	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_Germany	Gender_Female
Exited	1,00000	-0,02709	0,28532	-0,01400	0,11853	-0,04782	-0,00714	-0,15613	0,01210	0,17349	0,10651
CreditScore	-0,02709	1,00000	-0,00396	0,00084	0,00627	0,01224	-0,00546	0,02565	-0,00138	0,00554	0,00286
Age	0,28532	-0,00396	1,00000	-0,01000	0,02831	-0,03068	-0,01172	0,08547	-0,00720	0,04690	0,02754
Tenure	-0,01400	0,00084	-0,01000	1,00000	-0,01225	0,01344	0,02258	-0,02836	0,00778	-0,00057	-0,01473
Balance	0,11853	0,00627	0,02831	-0,01225	1,00000	-0,30418	-0,01486	-0,01008	0,01280	0,40111	-0,01209
NumOfProducts	-0,04782	0,01224	-0,03068	0,01344	-0,30418	1,00000	0,00318	0,00961	0,01420	-0,01042	0,02186
HasCrCard	-0,00714	-0,00546	-0,01172	0,02258	-0,01486	0,00318	1,00000	-0,01187	-0,00993	0,01058	-0,00577
IsActiveMember	-0,15613	0,02565	0,08547	-0,02836	-0,01008	0,00961	-0,01187	1,00000	-0,01142	-0,02049	-0,02254
EstimatedSalary	0,01210	-0,00138	-0,00720	0,00778	0,01280	0,01420	-0,00993	-0,01142	1,00000	0,01030	0,00811
Geography_Germany	0,17349	0,00554	0,04690	-0,00057	0,40111	-0,01042	0,01058	-0,02049	0,01030	1,00000	0,02463
Gender_Female	0,10651	0,00286	0,02754	-0,01473	-0,01209	0,02186	-0,00577	-0,02254	0,00811	0,02463	1,00000

3. Подготовка данных

Данные не требуют очистки, так как не выявлено пропусков, противоречий и экстремальных значений.

Разбиение на тестовую и обучающую выборки. Разбиение на тестовую и обучающую выборки производится случайным образом. В тестовую выборку пойдет 20% данных и в тренировочную 80%.

4. Моделирование

В расчете использовались следующие модели:

1. Логистическая регрессия

2. Дерево решений

3. Метод случайного леса

4. Градиентный бустинг

5. Наивный байесовский классификатор.

4.1. Логистическая регрессия.

Строим модель логистической регрессии:

```
Optimization terminated successfully.
         Current function value: 0.421980
         Iterations 6
Optimization terminated successfully.
         Current function value: 0.422032
         Iterations 6
                           Logit Regression Results
==============================================================================
Dep. Variable:                 Exited   No. Observations:                 8000
Model:                          Logit   Df Residuals:                     7991
Method:                           MLE   Df Model:                            8
Date:                Thu, 03 Nov 2022   Pseudo R-squ.:                  0.1592
Time:                        13:22:34   Log-Likelihood:                -3376.3
converged:                       True   LL-Null:                       -4015.7
Covariance Type:            nonrobust   LLR p-value:                8.937e-271
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                -3.6246      0.264    -13.727      0.000      -4.142      -3.107
CreditScore          -0.0010      0.000     -3.179      0.001      -0.002      -0.000
Age                   0.0720      0.003     24.923      0.000       0.066       0.078
Tenure               -0.0210      0.011     -1.984      0.047      -0.042      -0.000
Balance           2.706e-06   5.84e-07      4.631      0.000    1.56e-06    3.85e-06
NumOfProducts        -0.1080      0.053     -2.030      0.042      -0.212      -0.004
IsActiveMember       -1.1601      0.065    -17.739      0.000      -1.288      -1.032
Geography_Germany     0.7936      0.071     11.133      0.000       0.654       0.933
Gender_Female         0.5512      0.061      8.968      0.000       0.431       0.672
==============================================================================
```

Площадь под ROC – кривой составила 74%

4.2. Дерево решений.

Дерево решений построено со следующими параметрами:

Максимальная глубина (max_depth) = 8

Минимальное число объектов, при котором выполняется расщепление (min_samples_split) = 150

Максимальное количество листовых узлов (max_leaf_nodes) = 40

Модель дерева представлена на рисунке 5.

Площадь под ROC – кривой составила 85%

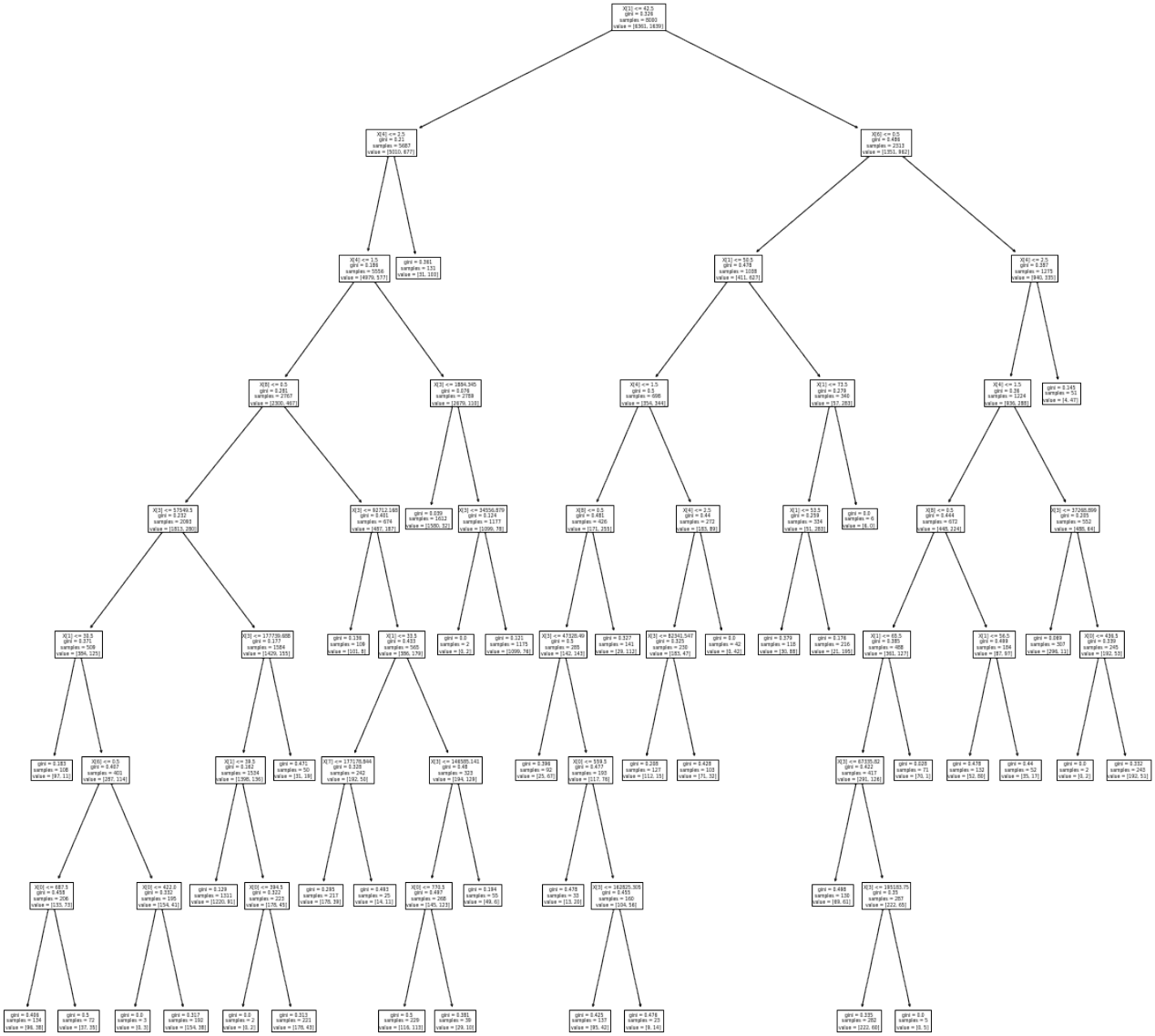

Рис.5 Дерево решений

4.3. Метод случайного леса.

Модель случайного леса построена со следующими параметрами:

Число деревьев (число оценок) (n_estimators) = 800

Число признаков для выбора расщепления (max_features) = 4

Максимальная глубина (max_depth) = 10

Площадь под ROC – кривой составила 86%

График значимости признаков для модели случайного леса представлен на рисунке 6.

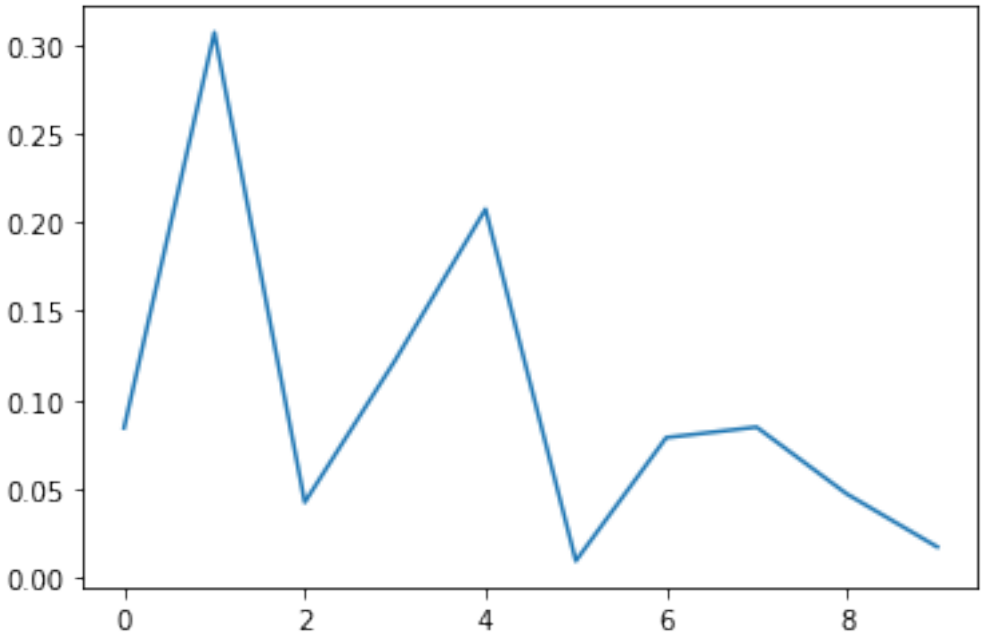

Рис.6 График значимости признаков для модели случайного леса

4.4. Градиентный бустинг.

Модель градиентного бустинга построена со следующими параметрами:

Число признаков для выбора расщепления (max_features) = 3,

Число оценок (n_estimators) = 500,

Скорость обучения (learning_rate) = 0.05,

Максимальная глубина (max_depth) = 4

Площадь под ROC – кривой составила 86%

4.5. Наивный байесовский метод.

Модель наивного байесовского метода построена со стандартными параметрами:

Площадь под ROC – кривой составила 75%

Для наглядности построим РОК-кривые для каждого метода.

РОК-кривые, построенные по моделям, представлены на рисунке 3.

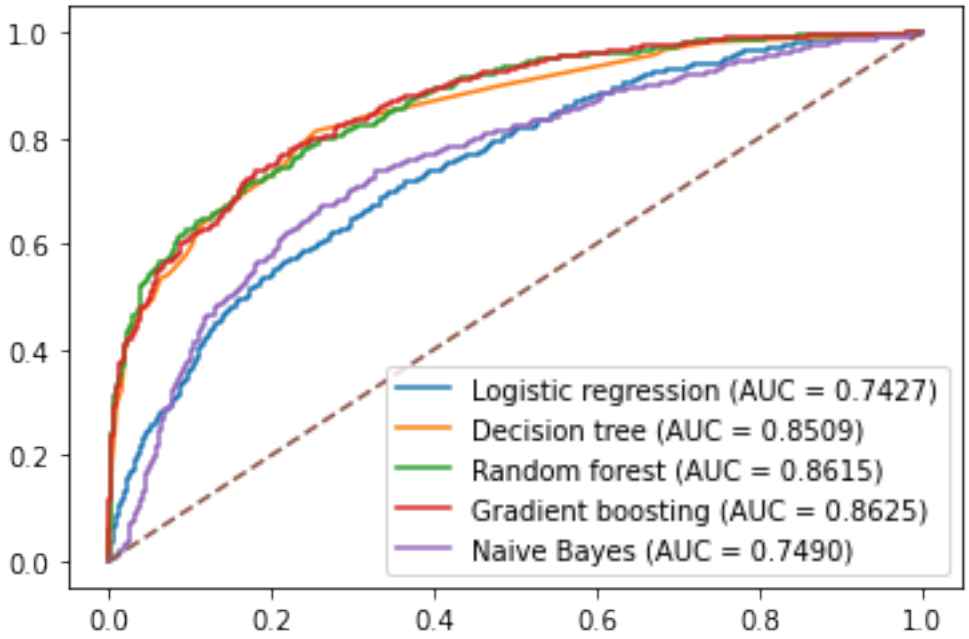

Рис. 3 ROK-кривые методов моделирования

5. Оценка результатов

Описательная точность результатов моделирования представлена в таблице 3:

Таблица 3. Точность результатов моделирования

№	Модель	Точность
1	Логистическая регрессия	0.7427
2	Дерево решений	0.8509
3	Метод случайного леса	0.8615
4	Градиентный бустинг	0.8625
5	Наивный байесовский классификатор	0.7490

Наиболее точными методами оценки являются Метод случайного леса и Градиентный бустинг.

Диаграмма точности результатов моделирования представлена на рисунке 4.

Рис. 4 Диаграмма точности результатов моделирования

6. Внедрение

Построенные модели могут с успехом применяться в деятельности коммерческих банков при разработке CRM-систем.

2. Приложение 1. Код программы Python для проведенного анализа

```
import numpy as np    # Импортируем библиотеку нампай
import pandas as pd   # Импортируем библиотеку пандас
import matplotlib.pyplot as plt   # Импортируем библиотеку матплот
либ
import statsmodels.api as sm # Импортируем библиотеку статсмодел
import sklearn               # Импортируем библиотеку склерн
from sklearn.model_selection import train_test_split
from google.colab import drive
drive.mount('/content/drive')


df = pd.read_csv('/content/drive/My Drive/Churn_Modelling.csv')  #
 Выводим исходные данные
df


df.dtypes  # выводим типы данных


df.isna().sum() # Проверяем наличие пропущенных значений


df.duplicated().sum() # Проверяем наличие дубликатов


df.columns[df.dtypes == 'object'] # Выводим названия колонк типа О
бъект


colm = df.columns  # находим столбцы с полностью одинаковыми значе
ниями
for i in colm:
  if len(df[i].unique()) == 1:
    print(i)


df = df.drop(['RowNumber',  'CustomerId', 'Surname'], axis = 1) #
удаляем не нужные для анализа столбцы
df


svodka = df.describe()
```

```
svodka.to_excel('/content/drive/My Drive/svodka.xlsx')  # Выводим сводку по датасету и сохраняем ее в файл Exel для дальнейшего использования в отчете
svodka


df.drop(['CreditScore', 'Age', 'Tenure', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'Exited', 'Geography', 'Gender'], axis = 1).boxplot(figsize = (12, 8))
       # Выводим диаграмму "Ящик с усами" для столбцов Balance и EstimatedSalary


df.drop(['Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited',
        'Exited', 'Geography', 'Gender'], axis = 1).boxplot(figsize = (12, 8))
        # Выводим диаграмму "Ящик с усами" для столбца CreditScore


df.drop(['Balance', 'EstimatedSalary', 'CreditScore', 'Age'], axis = 1).boxplot(figsize = (12, 8))
# Выводим диаграмму "Ящик с усами" для столбцов Tenure, NumOfProducts, HasCrCard, IsActiveMember, Exited


df.drop(['Balance', 'EstimatedSalary', 'CreditScore', 'Tenure', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'Exited', 'Geography', 'Gender'], axis = 1).boxplot(figsize = (12, 8))
        # Выводим диаграмму "Ящик с усами" для столбца Age


df['Age'].max()  # Находим максимальный возраст клиента


df = pd.get_dummies(df, drop_first = False) # Преобразуем категориальные переменные

df = df.drop('Gender_Male', axis = 1) # Удаляем столбец Gender_Male
df
```

```
df.columns # Выводим названия столбцов


df = df.reindex(columns = ['Exited', 'CreditScore', 'Age', 'Tenure
', 'Balance', 'NumOfProducts', 'HasCrCard',  # Переносим столбец '
Exited'
       'IsActiveMember', 'EstimatedSalary', 'Geography_France',
       'Geography_Germany', 'Geography_Spain', 'Gender_Female'])


train = df


train.corr().style.background_gradient(cmap = 'coolwarm') #строим
кореляционную матрицу


r = 0.5  # Находим столбцы с высокой корреляцией
col = []
cor = train.corr()
for k, i in enumerate(train.columns[0:-1]):
  for j in train.columns[k + 1:]:
    if np.abs(cor[i][j]) > r:
      if np.abs(cor['Exited'][j]) > np.abs(cor['Exited'][i]):
        col.append(i)
      else:
        col.append(j)
print(col)


train = train.drop(list(set(col)), axis = 1) # Удаляем стобцы с вы
сокой корреляцией


train.corr().style.background_gradient(cmap = 'coolwarm')  #Строим
 новую кореляционную матрицу


corr_matrix = train.corr().style.background_gradient(cmap = 'coolw
arm')  # Сохраняем корреляционную матрицу на диск
corr_matrix.to_excel('/content/drive/My Drive/corr_matrix.xlsx')
```

```
train_x, test_x, train_y, test_y = train_test_split(train.drop(['E
xited'], axis = 1), train['Exited'], test_size = 0.2) # Производим
 разбиение данных
model = sm.Logit(train_y, sm.add_constant(train_x)).fit()  # Строи
м модель логистической регрессии
print(model.summary())


# Создаем цикл для удаления из модели факторов, имеющих низкую зна
чимость
pvalue = 0.05
train1_x = train_x.copy()
test1_x = test_x.copy()
train1_y = train_y.copy()
test1_y = test_y.copy()

while np.max(model.pvalues[1:]) > pvalue:
  train1_x = train1_x.drop(train1_x.columns[np.argmax(model.pvalue
s[1:])], axis = 1)
  test1_x = test1_x.drop(test1_x.columns[np.argmax(model.pvalues[1
:])], axis = 1)
  model = sm.Logit(train1_y, sm.add_constant(train1_x)).fit()
print(model.summary())


pred_logit = model.predict(sm.add_constant(test1_x)) # Выводим пре
дсказания по модели


from sklearn.metrics import roc_auc_score, roc_curve  # импортируе
м функции для расчета ROC-кривой и площади под кривой
logit_roc_auc = roc_auc_score(test1_y, pred_logit)  # рассчитываем
 площадь под кривой
fpr, tpr, thresholds = roc_curve(test1_y, pred_logit)  # рассчитыв
аем fpr и tpr для построения ROC-кривой
plt.plot(fpr, tpr, label = 'Logistic regression (AUC = %0.4f)' %lo
git_roc_auc) # строим ROC-кривую
plt.plot([0, 1], [0, 1], linestyle = '--
') # строим диагональную линию для наглядности (это самый худший в
ариант для эффективности модели)
plt.legend() # выводим легенду на график
from sklearn import tree
```

```
model2 = tree.DecisionTreeClassifier(max_depth = 8, min_samples_sp
lit = 150, max_leaf_nodes = 40).fit(train_x, train_y)  # Строим мо
дель дерева решений
pred_tree = model2.predict_proba(test_x)
plt.figure(figsize = (20, 20))
tree.plot_tree(model2)


tree_roc_auc = roc_auc_score(test_y, pred_tree[:, 1]) # рассчитыва
ем площадь под кривой
fpr1, tpr1, thresholds1 = roc_curve(test_y, pred_tree[:, 1]) # рас
считываем fpr и tpr для построения ROK-кривой
plt.plot(fpr, tpr, label = 'Logistic regression (AUC = %0.4f)' %lo
git_roc_auc) # строим ROK-кривую
plt.plot(fpr1, tpr1, label = 'Decision tree (AUC = %0.4f)' %tree_r
oc_auc) # строим ROK-кривую
plt.plot([0, 1], [0, 1], linestyle = '--
') # строим диагональную линию для наглядности (это самый худший в
ариант для эффективности модели)
plt.legend() # выводим легенду на график


from sklearn import ensemble
model3 = sklearn.ensemble.RandomForestClassifier(max_features = 4,
 max_depth = 10, n_estimators = 800).fit(train_x, train_y)  # Стро
им модель Случайного леса
pred_rf = model3.predict_proba(test_x)[:, 1]


plt.plot(model3.feature_importances_)  # Построим график значимост
и признаков
len(model3.feature_importances_)


# Создадим цикл для удаления не значимых признаков
lst_index_for_del = []
lst =  model3.feature_importances_.tolist()
value = 0.01
for i in lst:
  if i > value:
   lst_index_for_del.append(lst.index(i))

print(lst_index_for_del)
```

```
train_x_2 = train_x[train_x.columns[lst_index_for_del]]
test_x_2 = test_x[test_x.columns[lst_index_for_del]]


model3_2 = sklearn.ensemble.RandomForestClassifier().fit(train_x_2
, train_y)  # Перестроим модель Случайного леса без учета не значи
мых признаков
pred_rf_2 = model3_2.predict_proba(test_x_2)[:, 1]


plt.plot(model3_2.feature_importances_)  # Построим график значимо
сти признаков без учета не значимых признаков
len(model3_2.feature_importances_)


rf_roc_auc = roc_auc_score(test_y, pred_rf)  # рассчитываем площад
ь под кривой
fpr2, tpr2, thresholds2 = roc_curve(test_y, pred_rf) # рассчитывае
м fpr и tpr для построения ROK-кривой
plt.plot(fpr, tpr, label = 'Logistic regression (AUC = %0.4f)' %lo
git_roc_auc) # строим ROK-кривую
plt.plot(fpr1, tpr1, label = 'Decision tree (AUC = %0.4f)' %tree_r
oc_auc)
plt.plot(fpr2, tpr2, label = 'Random forest (AUC = %0.4f)' %rf_roc
_auc)
plt.plot([0, 1], [0, 1], linestyle = '--
') # строим диагональную линию для наглядности (это самый худший в
ариант для эффективности модели)
plt.legend() # выводим легенду на график


# Построим ROC-
кривую, однако как видно из графика, модель с отобранными по значи
мости факторами показала результат хуже первоначальной модели
# Поэтому для дальнейшей работы оставим первоначальный вариант.
rf_roc_auc = roc_auc_score(test_y, pred_rf_2)
fpr2, tpr2, thresholds2 = roc_curve(test_y, pred_rf_2) # рассчитыв
аем fpr и tpr для построения ROK-кривой
plt.plot(fpr, tpr, label = 'Logistic regression (AUC = %0.4f)' %lo
git_roc_auc) # строим ROK-кривую
plt.plot(fpr1, tpr1, label = 'Decision tree (AUC = %0.4f)' %tree_r
oc_auc)
```

```
plt.plot(fpr2, tpr2, label = 'Random forest (AUC = %0.4f)' %rf_roc
_auc)
plt.plot([0, 1], [0, 1], linestyle = '--
') # строим диагональную линию для наглядности (это самый худший в
ариант для эффективности модели)
plt.legend() # выводим легенду на график


gb = sklearn.ensemble.GradientBoostingClassifier(max_features = 3,
 n_estimators = 500, learning_rate = 0.05, max_depth = 4).fit(trai
n_x, train_y)  # Строим модель градиентного бустинга
pred_gb = gb.predict_proba(test_x)[:, 1]


gb_roc_auc = roc_auc_score(test_y, pred_gb)  # рассчитываем площад
ь под кривой
fpr3, tpr3, thresholds3 = roc_curve(test_y, pred_gb) # рассчитывае
м fpr и tpr для построения ROK-кривой
plt.plot(fpr, tpr, label = 'Logistic regression (AUC = %0.4f)' %lo
git_roc_auc) # строим ROK-кривую
plt.plot(fpr1, tpr1, label = 'Decision tree (AUC = %0.4f)' %tree_r
oc_auc)
plt.plot(fpr2, tpr2, label = 'Random forest (AUC = %0.4f)' %rf_roc
_auc)
plt.plot(fpr3, tpr3, label = 'Gradient boosting (AUC = %0.4f)' %gb
_roc_auc)
plt.plot([0, 1], [0, 1], linestyle = '--
') # строим диагональную линию для наглядности (это самый худший в
ариант для эффективности модели)
plt.legend() # выводим легенду на график


from sklearn.naive_bayes import GaussianNB # Импортируем GaussianN
B
model4 = GaussianNB().fit(train_x, train_y) # Строим модель наивно
го байсовского метода
pred_nb = model4.predict_proba(test_x)[:, 1]


nb_roc_auc = roc_auc_score(test_y, pred_nb)  # рассчитываем площад
ь под кривой
fpr4, tpr4, thresholds4 = roc_curve(test_y, pred_nb) # рассчитывае
м fpr и tpr для построения ROK-кривой
```

```
plt.plot(fpr, tpr, label = 'Logistic regression (AUC = %0.4f)' %lo
git_roc_auc) # строим ROK-кривую
plt.plot(fpr1, tpr1, label = 'Decision tree (AUC = %0.4f)' %tree_r
oc_auc)
plt.plot(fpr2, tpr2, label = 'Random forest (AUC = %0.4f)' %rf_roc
_auc)
plt.plot(fpr3, tpr3, label = 'Gradient boosting (AUC = %0.4f)' %gb
_roc_auc)
plt.plot(fpr4, tpr4, label = 'Naive Bayes (AUC = %0.4f)' %nb_roc_a
uc)
plt.plot([0, 1], [0, 1], linestyle = '--
') # строим диагональную линию для наглядности (это самый худший в
ариант для эффективности модели)
plt.legend() # выводим легенду на график


results = pd.DataFrame()  # Выводим данные по предсказаниям
results['y'] = test_y
results['Logist_Regression'] = pred_logit
results['Decision_Tree'] = pred_tree[:, 1]
results['Random_Forest'] = pred_rf
results['Gradient_Boosting'] = pred_gb
results['Naive_Bayes'] = pred_nb
results.to_excel('/content/drive/My Drive/results.xlsx') # Записыв
аем полученные данные в файл ексель
results


# Строим диаграмму точности результатов моделирования
procent_tosnost = [logit_roc_auc, tree_roc_auc, rf_roc_auc, gb_roc
_auc, nb_roc_auc]
procent_tosnost = list(np.around(procent_tosnost, decimals = 4))
metod_analiza = ['Логистическая регрессия', 'Дерево решений', 'Мет
од случайного леса', 'Градиентный бустинг', 'Наивный байесовский м
етод']
plt.figure(figsize = (9, 7))
plt.bar(metod_analiza, procent_tosnost)
plt.title('Точность результатов моделирования', color = 'blue', fo
ntstyle = 'italic', family = 'arial', weight = 'bold', size = 'x-
large', fontsize = 20)
plt.xlabel('Метод', weight = 'bold', color = 'blue', fontsize = 15
)
```

```
plt.ylabel('Процент точности', weight = 'bold', color = 'blue', fo
ntsize = 15)
plt.xticks(rotation = 20)
plt.show()
```