УДК 004.896

Разработка прогностических моделей для предсказания ухода клиентов с использованием Python

Акентьев Данила Денисович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данной статьи является, анализ данных о клиентах банка и построение на их основе классификационных моделей. Программа написана в бесплатной интерактивной облачной среде для работы с кодом google collaboratory с ипользованием языка программирования Python и его библиотек pandas, numpy, matplotlib, seaborn, sklearn. Результатом исследования стали построенные классификационные модели для анализа и прогнозирования оттока клиентов.
Ключевые слова: анализ данных, отток клиентов, Python, google collaboratory, matplotlim, pandas, numpy, seaborn, sklearn

Development of predictive models for predicting customer care using Python

Akentev Danila Denisovich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this article is to analyze data on bank customers and build classification models based on them. The program is written in a free interactive cloud environment for working with google coollaboratory code using the Python programming language and its libraries pandas, numpy, matplotlib, seaborn, sklearn. The result of the study was the constructed classification models for analyzing and predicting customer churn.
Keywords: data analysis, customer churn, python, google collaboratory, matplotlib, pandas, numpy, seaborn, sklearn

1. Введение

1.1 Актуальность

Python - один из наиболее часто используемых языков программирования для анализа финансовых данных, с множеством полезных библиотек и встроенной функциональностью. В данной статье будет показано, как библиотеки машинного обучения Python можно использовать для прогнозирования оттока клиентов. Отток клиентов — это финансовый термин, который относится к потере клиента, то есть, когда клиент перестает взаимодействовать с компанией или бизнесом. Аналогичным образом,

скорость оттока — это скорость, с которой клиенты покидают компанию в течение определенного периода времени. Уровень оттока, превышающий определенный порог, может иметь как материальные, так и нематериальные последствия для успеха бизнеса компании. В идеале, компании хотели бы удержать как можно больше клиентов. С появлением передовых технологий обработки данных и машинного обучения компании теперь могут выявлять потенциальных клиентов, которые могут прекратить вести с ними бизнес в ближайшем будущем. В данной статье будет рассмотрен пример, как с помощью библиотек Python можно сделать первичный анализ данных.

1.2 Обзор исследований

В своей работе В.С. Мхитарян, М.Ю. Ахипова и Т.А. Дуброва описывают теоретические и алгоритмические основы анализа данных [1]. В.А. Мельникова, Д.А. Медведев рассказывают почему именно Python чаще всего используется в анализе данных и приводят этому ряд примеров и аргументов [2]. А.А. Карякина и А.В. Мельников сравнивают модели прогнозирования для оттока клиентов интернет-провайдеров [3]. Представили разработку алгоритма для увелечения точности прогноза потребности в готовой продукции в своей работе Е.В. Ценина и Е.В.Слепенкова [4]. Л.А. Ельшин, А.М. Гильманов и В.В. Бандеров разработали модель машинного обучения для прогнозирования курса криптовалюта на один день вперед с минимальным уровнем ошибок [5].

1.3 Цель исследования

Цель исследования – с помощью языка программирования Python и его библиотек для анализа данных проанализировать таблицу данных и построить классификационные модели прогнозирования.

2. Материалы и методы

Для написания программы используется среда программирования google colab [6] и таблица с данными [7].

3. Результаты и обсуждение

Перед началом работы следует зайти в свой аккаунт google colab и загрузить в него таблицу с данными. После чего импортируем нужные библиотеки, классификаторы, такие как: “К-ближайших соседей”, “Градиентный бустинг”, “Случайный лес” и “Логистическая регрессия”. Так же добавляем методы автоматической обработки данных и метрики качества классификации (рис.1).

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

from sklearn.neighbors import KNeighborsClassifier  #Импорт классификатора "К-ближайших соседей"
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier #Импорт классификатора "Градиентный бустинг" и "Рандомный лес"
from sklearn.linear_model import LogisticRegression  #Импорт классификатора "Логистическая регрессия"

#Импорт методов автоматической обработки данных
from sklearn.model_selection import GridSearchCV, train_test_split, StratifiedKFold

#Импорт метрик качества классификации
from sklearn.metrics import recall_score, precision_score, accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
```

Рисунок 1 - Импорт нужных библиотек

Загружаем данные с помощью pandas (рис.2).

```
df= pd.read_csv('Churn.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Рисунок 2 - Загрузка данных

Удаляем лишние данные, такие как номер строки, ID клиентов и фамилии, так как они не имеют значения при анализе (рис.3).

```
df=df.drop(['RowNumber', 'CustomerId','Surname'], axis=1)
df
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0
...	...	...	...	...	...	...	...	...	...	...	...
9995	771	France	Male	39	5	0.00	2	1	0	96270.64	0
9996	516	France	Male	35	10	57369.61	1	1	1	101699.77	0
9997	709	France	Female	36	7	0.00	1	0	1	42085.58	1
9998	772	Germany	Male	42	3	75075.31	2	1	0	92888.52	1
9999	792	France	Female	28	4	130142.79	1	1	0	38190.78	0

10000 rows × 11 columns

Рисунок 3 - Удаление лишних данных

Нужно выяснить какие данные оказывают наибольшее влияние на отток клиентов. Для этого строим гистограммы с различными параметрами. Исходя из графиков видно, что наибольшее влияние оказывают активность, количество продуктов и возраст (рис.4).

```
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(24, 8))
for i in range(2):
  for j in range(4):
    sns.countplot(data=df,x=df.columns[j+i*4],hue='Exited',ax=axes[i][j])
```

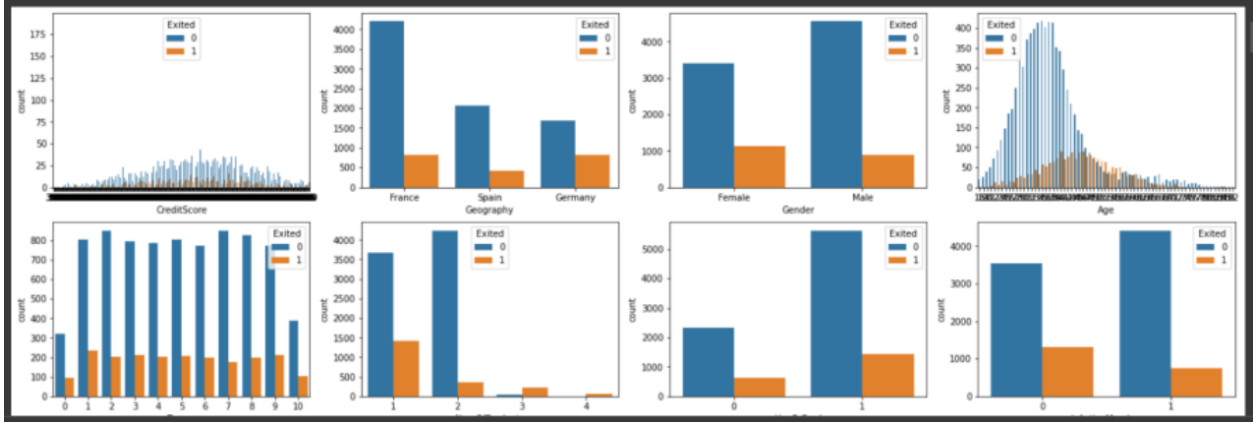


Рисунок 4 - Первичный анализ данных

Далее разделяем данные на целевые и влияющие на целевой столбец (рис.5).

```
y = df['Exited']
x = df[['Age','NumOfProducts','IsActiveMember']]
```

Рисунок 5 - Разделение данных

Заготавливаем основы для нахождения лучших параметров, создаваемых моделей (рис.6).

```
knn_params = {'n_neighbors' : np.arange(1, 10, 1)}  # Параметры для классификатора KNeighborsClassifier

gbc_params = {'learning_rate': np.arange(0.1, 0.6, 0.1)}   # Параметры для классификатора GradientBoostingClassifier

rfc_params = {'n_estimators': range(10, 100, 10),  # Параметры для классификатора RandomForestClassifier
              'min_samples_leaf': range(1, 7)}

svc_params = {'kernel': ['linear', 'rbf'],
'C': np.arange(0.1, 1, 0.2)}                       # Параметры для классификатора SVC

lr_params = {'C': np.arange(0.2, 1, 0.1)} # Параметры для классификатора LogisticRegression

skf = StratifiedKFold(n_splits=8, random_state=17, shuffle=True) # Параметры для кросс-валидации
```

Рисунок 6 - Подготовка моделей

Разделяем данные на тренировочные и тестовые (рис.7).

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,  stratify=y, random_state=24)
```

Рисунок 7 - Подготовка данных

Переходим к созданию объектов моделей, определению для них лучшего параметра и обучению на основе данных (рис.8).

```
knn = KNeighborsClassifier()                    # Определение объекта классификатора KNeighborsClassifier
gscv_knn = GridSearchCV(estimator=knn, param_grid=knn_params, cv=skf)  # Определение объекта кросс-валидации для KNeighborsClassifier
knn_model = gscv_knn.fit(X_train, y_train)  # Обучение модели KNeighborsClassifier на кросс-валидации

gbc = GradientBoostingClassifier(random_state=17)  # Определение объекта классификатора GradientBoostingClassifier
gscv_gbc = GridSearchCV(estimator=gbc, param_grid=gbc_params, cv=skf)  # Определение объекта кросс-валидации для GradientBoostingClassifier
gbc_model = gscv_gbc.fit(X_train, y_train)  # Обучение модели GradientBoostingClassifier на кросс-валидации

rfc = RandomForestClassifier(random_state=17)      # Определение объекта классификатора RandomForestClassifier
gscv_rfc = GridSearchCV(estimator=rfc, param_grid=rfc_params, cv=skf)  # Определение объекта кросс-валидации для RandomForestClassifier
rfc_model = gscv_rfc.fit(X_train, y_train)  # Обучение модели RandomForestClassifier на кросс-валидации

lr = LogisticRegression(random_state=17,
                        #class_weight = {1:5},
                        solver = 'liblinear')      # Определение объекта классификатора LogisticRegression

gscv_lr = GridSearchCV(estimator=lr, param_grid=lr_params, cv=skf)     # Определение объекта кросс-валидации для LogisticRegression
lr_model = gscv_lr.fit(X_train, y_train)    # Обучение модели LogisticRegression на кросс-валидации
```

Рисунок 8 - Создание и обучение моделей

Отобразим лучшие параметры, которые были выбраны для каждой модели (рис. 9).

```
gscv_knn.best_params_
{'n_neighbors': 9}

gscv_gbc.best_params_
{'learning_rate': 0.1}

gscv_rfc.best_params_
{'min_samples_leaf': 6, 'n_estimators': 10}

gscv_lr.best_params_
{'C': 0.2}
```

Рисунок 9 - Лучшие параметры для каждой модели

Осталось проверить модели тестовыми данными. Для этого приводим их к удобному виду (рис.10).

```
knn_predict = knn_model.predict(X_test) #"К-ближайших соседей"
gbc_predict = gbc_model.predict(X_test) #Градиентный бустинг
lr_predict = lr_model.predict(X_test) #линейная ригрессия
rfc_predict = rfc_model.predict(X_test) #Рандомный лес

models_names = ['KNeighbors', 'GradientBoosting', 'RandomForest', 'LogisticRegression'] #Все модели
predicts = [knn_predict, gbc_predict, rfc_predict, lr_predict]
```

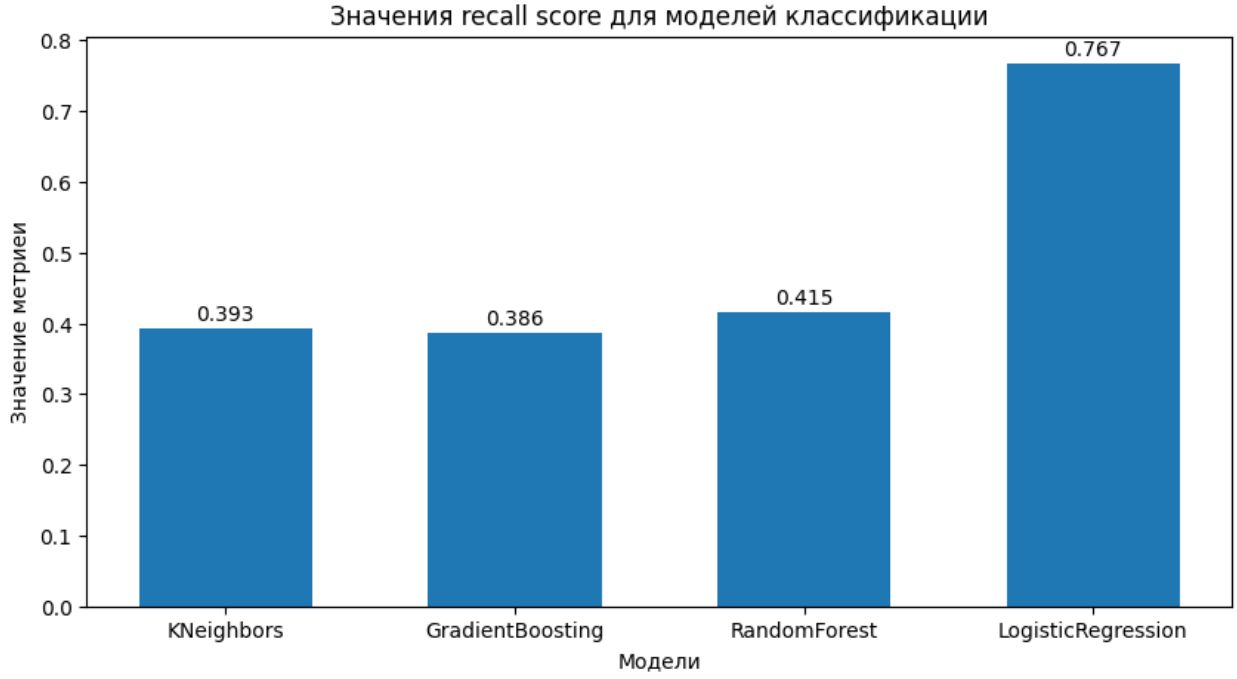
Рисунок 10 - Подготовка моделей к тестированию

Приступаем к проверке созданных моделей с помощью метрик, таких как recall score - полнота, precision score - точность, accuracy score - доля правильных ответов алгоритма (рис.11).

```python
# Получение значений полноты моделей классификаторов
metrics_scores = [recall_score, precision_score, accuracy_score]
scores_names = ['recall_score', 'precision_score', 'accuracy_score']
values_list = []
for i, score in enumerate(metrics_scores):
    for predict in predicts:
        values_list.append(round(score(y_test, predict),3))
    x = np.arange(len(models_names))  # Список координат столбцов по x
    fig, ax = plt.subplots(figsize=(10,5))  # Определение фигуры и осей
    rects = ax.bar(x, values_list, 0.6)  # Определение колонок для данных
    # Указание подписей для осей, таблицы, легенды
    ax.set_ylabel('Значение метрики')
    ax.set_xlabel('Модели')
    ax.set_title(f'Значения {scores_names[i]} для моделей классификации')
    ax.set_xticks(x)
    ax.set_xticklabels(models_names)
    # Определение функции для отображения столбцов с аннотациями
    def autolabel(rects):
        for rect in rects:
            height = rect.get_height()
            # Задание параметров для аннотаций
            ax.annotate('{}'.format(height),
                        xy=(rect.get_x() + rect.get_width() / 2, height), # Получение точек координат для текста
                        xytext=(0, 2),    # Высота текста над столбцами
                        # Расположение текста относительно столбцов
                        textcoords="offset points",
                        ha='center', va='bottom')
    autolabel(rects)  # Выполнение функции
    values_list = []
plt.show()
```

Рисунок 11 – Проверка моделей метриками

Выполнив проверку, получаем результаты (таб.1), из которых видно, что наиболее точными являются модели - Random Forest и Logistic Regression. Однако стоит заметить, что разница между всеми моделями не столь значительна (рис.12).



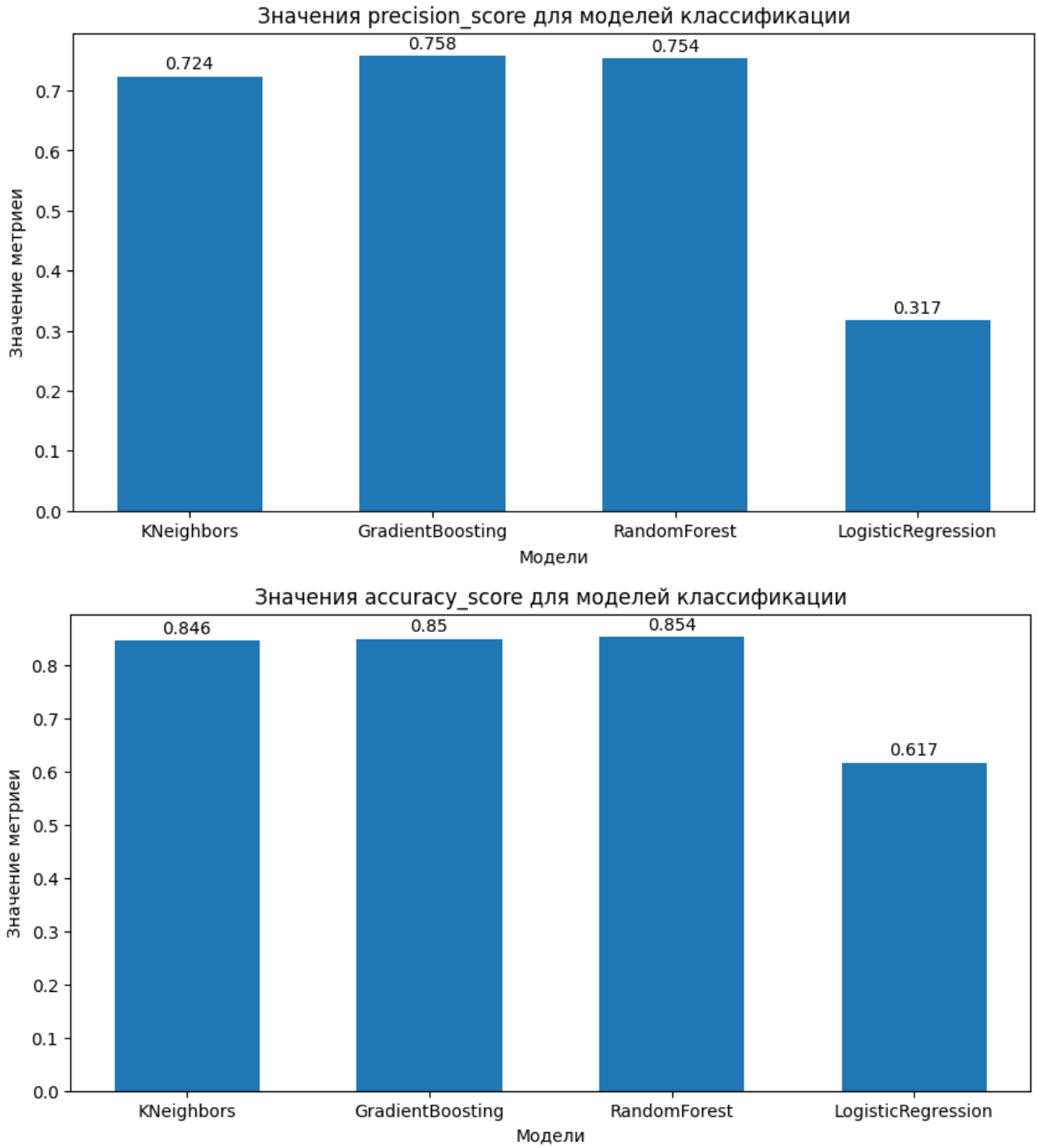


Рисунок 12 - Оценка метрик

Таблица 1. Результаты

	Recall score	Precision score	Accuracy score
KNeighbors	0.393	0.724	0.846
GradientBoosting	0.386	0.758	0.85
RandomForest	0.415	0.754	0.854
LogisticRegression	0.767	0.317	0.617

1. KNeighbors

- Recall: 0.393
- Precision: 0.724

- Accuracy: 0.846

Модель KNeighbors имеет средние значения полноты и точности, а также высокую общую точность. Это означает, что модель хорошо справляется с общими предсказаниями, но может пропускать значительное количество положительных примеров.

2. GradientBoosting
- Recall: 0.386
- Precision: 0.758
- Accuracy: 0.85

GradientBoosting демонстрирует схожие результаты с KNeighbors, но с немного более высокой точностью и общей точностью. Модель также может пропускать много положительных примеров, несмотря на высокую точность.

3. RandomForest
- Recall: 0.415
- Precision: 0.754
- Accuracy: 0.854

Модель RandomForest показывает лучшую полноту среди первых трех моделей, что означает, что она лучше идентифицирует положительные примеры. Точность и общая точность также находятся на высоком уровне, что делает эту модель наиболее сбалансированной из первых трех по представленным метрикам.

4. LogisticRegression
- Recall: 0.767
- Precision: 0.317
- Accuracy: 0.617

LogisticRegression имеет значительно более высокую полноту, что указывает на способность модели идентифицировать большинство положительных примеров. Однако точность очень низкая, что означает высокое количество ложноположительных предсказаний. Общая точность также ниже по сравнению с другими моделями.

Выводы

Благодаря Python и сопутствующим библиотекам по анализу данных, удалось проанализировать данные клиентов и построить на их основе классификационные модели для прогнозирования оттока клиентов.

RandomForest: Наиболее сбалансированная модель с хорошей полнотой, точностью и общей точностью.

LogisticRegression: Подходит для задач, где критична высокая полнота (например, обнаружение заболеваний), несмотря на низкую точность и общую точность.

KNeighbors и GradientBoosting: Хороши для задач, где важна высокая общая точность и приемлемая точность предсказаний, но могут пропускать положительные примеры.

В зависимости от конкретных требований задачи можно выбрать соответствующую модель. Если критично не пропустить положительные примеры, стоит рассмотреть LogisticRegression или RandomForest. Если важны сбалансированные показатели и высокая общая точность, лучше подойдут RandomForest, KNeighbors или GradientBoosting.

Библиографический список

1. Мхитарян В.С. Анализ данных в MS Excel. М.: КУРС, 2019. 368 с.
2. Мельникова В. А., Медведев Д. А. Анализ больших данных с использованием Python //Труды Братского государственного университета. Серия: Естественные и инженерные науки. 2019. Т. 1. С. 46-49.
3. Карякина А. А., Мельников А. В. Сравнение моделей прогнозирования оттока клиентов интернет-провайдеров // Машинное обучение и анализ данных. 2017. Т. 3. №. 4. С. 250-256.
4. Ценина Е.В., Слепенкова Е.В. Разработка алгоритма машинного обучения для прогнозирования потребности в запасах готовой продукции // Риск: ресурсы, информация, снабжение, конкуренции. 2023. №. 3. С. 57-65.
5. Ельшин Л.А., Гильманов А.М., Бандеров В.В. Разработка прогноза динамики курса криптовалюты на основе теории машинного обучения // Финансовая аналитика: проблемы и решения. 2020. Т. 13. №. 1(351). С. 97-113.
6. URL: https://colab.google.com/
7. URL: https://drive.google.com/file/d/1fHdf8SKnH-mMV1I_Miljc88z2jj3axLA/view?usp=sharing