|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт информационных технологий |
| Кафедра корпоративных информационных систем |

|  |
| --- |
| **Индивидуальный проект**  На тему «Прогнозирование оттока клиентов телеком-оператора» |

|  |  |  |
| --- | --- | --- |
| **Выполнил:** | аспирант группы  КСАО-01-19 | Груненков Д.В. |
| **Преподаватель:** |  | Александров Д.В.,  д.т.н., проф. |

Москва 2021

**Оглавление**

[Введение 3](#_Toc91422381)

[1. Цель и постановка задачи 4](#_Toc91422382)

[2. Предлагаемый подход к решению задачи 6](#_Toc91422383)

[2.1. Загрузка и предобработка датасета 6](#_Toc91422384)

[2.2. Используемые методы классификации. 10](#_Toc91422385)

[2.2.1. Метод](#_Toc91422386) *[k](#_Toc91422386)*[-ближайших соседей. 10](#_Toc91422386)

[2.2.2. Решающее дерево 11](#_Toc91422387)

[3. Реализация и результаты 12](#_Toc91422388)

[4. Оценка качества результатов и заключение 16](#_Toc91422389)

[Список использованных источников 17](#_Toc91422390)

# Введение

С проблемой оттока клиентов сталкиваются все компании, предоставляющие товары или услуги. В связи с высокой стоимостью привлечения новых пользователей и ростом конкурентоспособности рынка увеличивается значимость удержания клиентов. Поэтому современные компании заинтересованы в создании инструментов для более гибкой работы с пользователями, склонных перейти к конкурентам.

Основной целью работы является исследование методов машинного обучения и построение прогнозирующей модели оттока клиентов в телекоммуникационной компании.

# Цель и постановка задачи

Одна из типичных задач, возникающих в процессе определения поведения абонента — это определение принадлежности клиента к одному из двух классов: лояльных к компании и склонных к уходу (задача бинарной классификации). С помощью современных методов машинного обучения и первичной обработки данных эта задача эффективно решается.

Для обучения моделей будем использовать набор данных telecom\_churn [1]. Каждая строка базы представляет собой информацию по одному клиенту – это объект исследования. Столбцы – признаки объекта. Описание признаков приведено в таблице 1.

Таблица 1. Описание признаков.

| **Название** | **Описание** | **Тип** |
| --- | --- | --- |
| **State** | Буквенный код штата | номинальный |
| **Account length** | Как долго клиент обслуживается компанией | количественный |
| **Area code** | Префикс номера телефона | количественный |
| **International plan** | Международный роуминг (подключен/не подключен) | бинарный |
| **Voice mail plan** | Голосовая почта (подключена/не подключена) | бинарный |
| **Number vmail messages** | Количество голосовых сообщений | количественный |
| **Total day minutes** | Общая длительность разговоров днем | количественный |
| **Total day calls** | Общее количество звонков днем | количественный |
| **Total day charge** | Общая сумма оплаты за услуги днем | количественный |
| **Total eve minutes** | Общая длительность разговоров вечером | количественный |
| **Total eve calls** | Общее количество звонков вечером | количественный |
| **Total eve charge** | Общая сумма оплаты за услуги вечером | количественный |
| **Total night minutes** | Общая длительность разговоров ночью | количественный |
| **Total night calls** | Общее количество звонков ночью | количественный |
| **Total night charge** | Общая сумма оплаты за услуги ночью | количественный |
| **Total intl minutes** | Общая длительность международных разговоров | количественный |
| **Total intl calls** | Общее количество международных разговоров | количественный |
| **Total intl charge** | Общая сумма оплаты за международные разговоры | количественный |
| **Customer service calls** | Число обращений в сервисный центр | количественный |

Целевая переменная: **Churn** – Признак оттока, бинарный признак (1 – потеря клиента, то есть отток). В работе будем строить модели, прогнозирующие этот признак по остальным.

# Предлагаемый подход к решению задачи

Одной из главных характеристик классификатора является его эффективность. Очевидный способ оценивания эффективности — доля правильных ответов.



где P — количество объектов, чей класс совпал с классом, определенным алгоритмом, а N — размер тестовой выборки.

# Загрузка и предобработка датасета

*# импорт библиотек****import*** *numpy* ***as*** *np****import*** *pandas* ***as*** *pd****from*** *matplotlib* ***import*** *pyplot* ***as*** *plt*

Считаем данные из файла и выведем первые пять строк.

*# чтение данных из файла  
df = pd.read\_csv(****'telecom\_churn.csv'****)  
# выведем первые 5 строк  
df.head(5)*

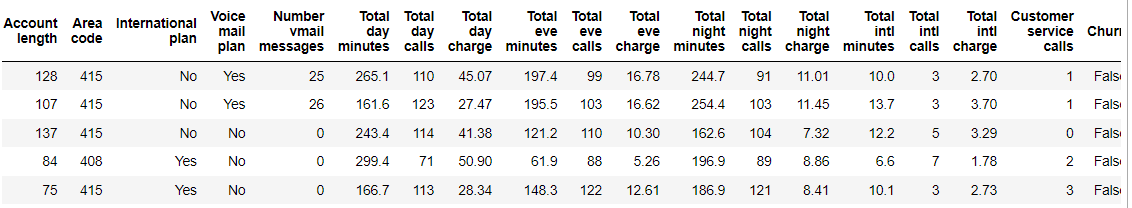


Рисунок 1 – Первые 5 строк таблицы.

*# выведем размер датасета  
print(df.shape)*

В таблице 3333 строки и 20 столбцов.

*# выведем названия столбцов и краткую информацию по каждому столбцу  
print(df.columns)  
print(df.info())*

*Index(['State', 'Account length', 'Area code', 'International plan',*

*'Voice mail plan', 'Number vmail messages', 'Total day minutes',*

*'Total day calls', 'Total day charge', 'Total eve minutes',*

*'Total eve calls', 'Total eve charge', 'Total night minutes',*

*'Total night calls', 'Total night charge', 'Total intl minutes',*

*'Total intl calls', 'Total intl charge', 'Customer service calls',*

*'Churn'],*

*dtype='object')*

*<class 'pandas.core.frame.DataFrame'>*

*RangeIndex: 3333 entries, 0 to 3332*

*Data columns (total 20 columns):*

*State 3333 non-null object*

*Account length 3333 non-null int64*

*Area code 3333 non-null int64*

*International plan 3333 non-null object*

*Voice mail plan 3333 non-null object*

*Number vmail messages 3333 non-null int64*

*Total day minutes 3333 non-null float64*

*Total day calls 3333 non-null int64*

*Total day charge 3333 non-null float64*

*Total eve minutes 3333 non-null float64*

*Total eve calls 3333 non-null int64*

*Total eve charge 3333 non-null float64*

*Total night minutes 3333 non-null float64*

*Total night calls 3333 non-null int64*

*Total night charge 3333 non-null float64*

*Total intl minutes 3333 non-null float64*

*Total intl calls 3333 non-null int64*

*Total intl charge 3333 non-null float64*

*Customer service calls 3333 non-null int64*

*Churn 3333 non-null bool*

*dtypes: bool(1), float64(8), int64(8), object(3)*

bool, int64, float64 и object — это типы признаков. 1 признак — логический (bool), 3 признака имеют тип object и 16 признаков — числовые. Также с помощью метода info удобно быстро посмотреть на пропуски в данных, в нашем случае их нет, в каждом столбце по 3333 наблюдения.

Применим метод astype к признаку Churn и переведём его в int64:

*df[****'Churn'****] = df[****'Churn'****].astype(****'int64'****)*

Для категориальных (тип object) и булевых (тип bool) признаков можно воспользоваться методом value\_counts. Посмотрим на распределение данных по нашей целевой переменной — Churn:

*df[****'Churn'****].value\_counts()*

*0 2850*

*1 483*

*Name: Churn, dtype: int64*

2850 пользователей из 3333 — лояльные, значение переменной Churn у них — 0. Узнаем, какова доля людей нелояльных пользователей в нашем датафрейме?

*df[****'Churn'****].mean()*

*Результат - 0.14491449144914492.*

Это означает, что отток клиентов компании составляет 14.5%.

Проведем предобработку. Удалим из датафрейма столбец с обозначением штата, преобразуем категориальные признаки в числовые, а целевой столбец Churn выделим в отдельную переменную.

*# заменим категориальные признаки числовыми  
df[****'International plan'****] = pd.factorize(df[****'International plan'****])[0]  
df[****'Voice mail plan'****] = pd.factorize(df[****'Voice mail plan'****])[0]  
# приведем целевую переменную к числовому виду  
df[****'Churn'****] = df[****'Churn'****].astype(****'int'****)  
# отдельно скопируем стобик с названием штата  
states = df[****'State'****]  
# отдельно выделим целевую переменную  
y = df[****'Churn'****]  
# удалим из датасета стобцы с названием штата и целевую переменную  
df.drop([****'State'****,* ***'Churn'****], axis=1, inplace=****True****)*

Теперь датасет состоит только из числовых признаков, целевая переменная отделена от датасета и можно приступать к тренировке моделей.

# Используемые методы классификации.

В настоящее время применяется огромное количество алгоритмов машинного обучения для задачи классификации. В нашей работе были выбраны метод k-ближайших соседей и решающие деревья.

## Метод *k*-ближайших соседей.

Задана обучающая выборка — множество объектов, Y — множество классов, а m — мощность T. На X введена функция расстояния. Чем больше значение этой функции, тем менее схожими являются объекты  и .

Для произвольного объекта x ∈ X расположим объекты обучающей выборки xi в порядке возрастания расстояний до x:



где  — объект обучающей выборки, который является i-м соседом объекта x. Аналогичное обозначение вводится и для классов на i-м соседе . Каждый x ∈ X порождает свою нумерацию. Из этих упорядоченных элементов T выбираются первые k (задающийся параметр), а затем x присваивается тот класс, который преобладает среди этих k элементов.

В общем виде алгоритм k-ближайших соседей можно записать следующим образом [2]:



где  — заданная весовая функция, оценивающая насколько сильно i-й сосед влияет на объект x.

## Решающее дерево

Дерево принятия решений [3] — это алгоритм классификации, работающий над деревом (в нашем случае бинарным), в котором каждой внутренней вершине приписан предикат , каждой терминальной вершине приписано имя класса . При классификации объекта x ∈ X он проходит по дереву путь от корня до некоторого листа.

Объект x доходит до вершины  тогда и только тогда, когда выполняется конъюнкция , составленная из всех предикатов, приписанных внутренним вершинам дерева на пути от корня до вершины . Множества объектов  попарно не пересекаются, а их объединение совпадает со всем пространством X.

Отсюда следует, что алгоритм классификации , реализуемый бинарным решающим деревом, можно записать в виде простого голосования конъюнкций [11]:



причем для любого x ∈ X одно и только одно слагаемое во всех этих суммах равно единице.

Требование максимизации информативности конъюнкций  означает, что каждая из них должна выделять как можно больше обучающих объектов, допуская при этом как можно меньше ошибок. Число листьев в дереве должно быть как можно меньше, и они должны покрывать части выборки примерно одинаковой мощности.

# Реализация и результаты

Представленные выше алгоритмы возможно реализовать на любом алгоритмическом языке, однако в настоящее время Python предоставляет огромное количество готовых библиотек с реализованными алгоритмами машинного обучения. Одна из таких библиотек - Scikit Learn.

Scikit-learn - один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он позволяет выполнять множество операций и предоставляет множество алгоритмов. Scikit-learn также предлагает отличную документацию о своих классах, методах и функциях, а также описание используемых алгоритмов.

Scikit-Learn поддерживает: предварительную обработку данных; уменьшение размерности; выбор модели; алгоритмы регрессии; алгоритмы классификации; кластерный анализ [3, 4].

Опишем ход решения задачи предсказания оттока пользователей с помощью классов sklearn.

Выделим 70% выборки (X\_train, y\_train) под обучение и 30% будут тестовой выборкой (X\_holdout, y\_holdout). Тестовая выборка никак не будет участвовать в настройке параметров моделей, на ней мы в конце, после этой настройки, оценим качество полученной модели. Обучим 2 модели – дерево решений и kNN, пока не знаем, какие параметры хороши, поэтому наугад: глубину дерева берем 4, число ближайших соседей – 7.

*# импорт нужных функций****from*** *sklearn.model\_selection* ***import*** *train\_test\_split, StratifiedKFold****from*** *sklearn.neighbors* ***import*** *KNeighborsClassifier****from*** *sklearn.tree* ***import*** *DecisionTreeClassifier  
  
# разделение на ренировочный и тестовый набор  
X\_train, X\_holdout, y\_train, y\_holdout = train\_test\_split(df.values, y, test\_size=0.3,  
random\_state=17)  
  
# создание двух класссификаторовж  
tree = DecisionTreeClassifier(max\_depth=5, random\_state=17)  
knn = KNeighborsClassifier(n\_neighbors=10)  
  
# тренировка моделей  
tree.fit(X\_train, y\_train)  
knn.fit(X\_train, y\_train)*

Качество прогнозов будем проверять с помощью простой метрики – доли правильных ответов. Сделаем прогнозы для отложенной выборки. Дерево решений справилось лучше: доля правильных ответов около 92% против 87.7% у kNN. Но это мы пока выбирали параметры наугад.

***from*** *sklearn.metrics* ***import*** *accuracy\_score  
  
tree\_pred = tree.predict(X\_holdout)  
accuracy\_score(y\_holdout, tree\_pred) # 0.92  
  
knn\_pred = knn.predict(X\_holdout)  
accuracy\_score(y\_holdout, knn\_pred) # 0.877*

Теперь настроим параметры дерева на кросс-валидации. Настраивать будем максимальную глубину и максимальное используемое на каждом разбиении число признаков. Суть того, как работает GridSearchCV: для каждой уникальной пары значений параметров max\_depth и max\_features будет проведена 5-кратная кросс-валидация и выберется лучшее сочетание параметров.

***from*** *sklearn.model\_selection* ***import*** *GridSearchCV, cross\_val\_score  
  
tree\_params = {****'max\_depth'****: range(1,11),****'max\_features'****: range(4,19)}  
  
tree\_grid = GridSearchCV(tree, tree\_params,  
cv=5, n\_jobs=-1,  
verbose=****True****)  
  
tree\_grid.fit(X\_train, y\_train)*

Лучшее сочетание параметров и соответствующая средняя доля правильных ответов на кросс-валидации показаны на рисунке 2. Как видим – глубина дерева составляет 6 единиц, а максимальное количество признаков – 17.

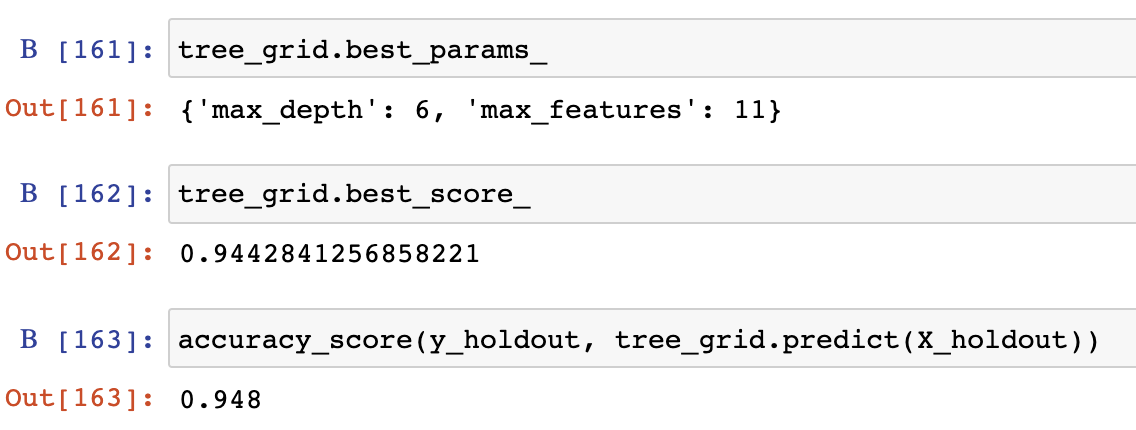


Рисунок 2 – Настройка решающего дерева

Теперь попробуем настроить число соседей в алгоритме kNN. Также перед запуском алгоритма применим к данным масштабирование.

***from*** *sklearn.pipeline* ***import*** *Pipeline****from*** *sklearn.preprocessing* ***import*** *StandardScaler****from*** *sklearn.pipeline* ***import*** *Pipeline****from*** *sklearn.preprocessing* ***import*** *StandardScaler  
  
knn\_pipe = Pipeline([(****'scaler'****, StandardScaler()), (****'knn'****, KNeighborsClassifier(n\_jobs=-1))])  
  
knn\_params = {****'knn\_\_n\_neighbors'****: range(1, 10)}  
  
knn\_grid = GridSearchCV(knn\_pipe, knn\_params,  
cv=5, n\_jobs=-1,  
verbose=****True****)  
  
knn\_grid.fit(X\_train, y\_train)  
  
knn\_grid.best\_params\_, knn\_grid.best\_score\_*

Наилучшие результаты дает модель с 7 соседями, что показано на рисунке 3.

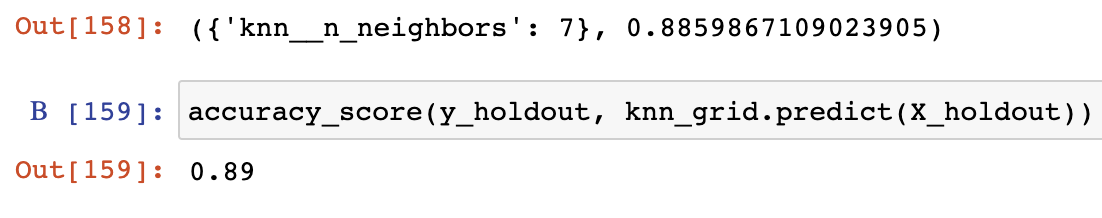


Рисунок 3 – Настройка параметров kNN

# Оценка качества результатов и заключение

В ходе работы мы поставили задачу предсказания нелояльных пользователей телеком-оператора, описали набор данных и числовые признаки каждого клиента, провели предобработку данных с помощью библиотеки pandas, изучили два алгоритма машинного обучения (KNN, решающие деревья), реализовали их на практике с помощью библиотеки sklearn и применили к поставленной задаче.

В этом примере дерево показало себя лучше, чем метод ближайших соседей: 92% правильных ответов на кросс-валидации и 94.8% на отложенной выборке против 87.7% / 89% для kNN.

# Список использованных источников

1. Набор данных об оттоке клиентов. Эл. ресурс. Режим доступа: <https://www.kaggle.com/becksddf/churn-in-telecoms-dataset>

2. Статья "Метод ближайших соседей". Эл. ресурс. Режим доступа: <http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9>

3. Орельен Жерон Прикладное машинное обучение с помощью Scikit-Learn, Keras и TensorFlow. Концепции, инструменты. // Изд.: Диалектика, 2020 г., С. 1040

4. Статья «Что такое Scikit Learn». Эл. ресурс. Режим доступа: <https://datastart.ru/blog/read/chto-takoe-scikit-learn-gayd-po-populyarnoy-biblioteke-python-dlya-nachinayuschih>