Федеральное государственное образовательное бюджетное
учреждение высшего профессионального образования
**«Финансовый университет при Правительстве
Российской Федерации»**

Департамент «Департамент анализа данных, принятия решений и финансовых технологий»

Курсовая работа
на тему:

«Предварительный анализ данных и построение признаков в задачах предсказания оттока клиентов»

Вид исследуемых данных: «Telco Customer Churn»

Выполнил:

Студент группы ПМ18-2
факультета «Прикладная математика и анализ больших данных»
Поздняков А.Р.

Научный руководитель:

доцент, к.э.н. Медведев А.В.

Москва, 2020

Оглавление

1. Введение

Приобретение и удержание клиентов-это самые главные проблемы в современном деловом мире. Стремительный рост рынка в каждом бизнесе ведет к увеличению абонентской базы. Следовательно, компании осознали важность удержания под рукой клиентов. Для поставщиков услуг стало обязательным снизить уровень оттока, поскольку небрежность может привести к снижению прибыльности в значительной перспективе. Прогнозирование оттока помогает выявить тех клиентов, которые, скорее всего, покинут компанию. Методы интеллектуального анализа данных позволяют компаниям быть оснащенными эффективными методами снижения уровня оттока. Главная проблема: удержание клиента в среднем в 2-5 раз дешевле, чем привлечение нового.

Отток клиентов - это склонность клиента покинуть поставщика услуг. Прогнозирование оттока клиентов-это процесс выявления тех клиентов, которые могут уйти или перейти из текущей компании-поставщика услуг по определенным причинам. Основная цель модели прогнозирования оттока состоит в том, чтобы определить таких клиентов, чтобы стратегии удержания могли быть нацелены на них, и компания могла процветать, максимизируя свой общий доход. Прогнозирование оттока клиентов было поднято как пресловутый вопрос во многих областях, таких как электросвязи, кредитные карты, интернет поставщики услуг, электронная коммерция, розничный маркетинг, газетные издательства, банковское дело и финансовые услуги.

Цель данной курсовой работы – изучить возможные методы обработки данных и предсказания оттока клиентов. Построить модели машинного обучения и сравнить их эффективность. Выбрать наилучшую и попытаться повысить её точность с помощью подбора гиперпараметров.

В этой работе будут описаны основные понятия оттока клиентов, различные метрики точности предиктивных моделей, предварительный анализ и обработка данных, описание и построение моделей машинного обучения.

Производиться все вышеупомянутые действия будут на датасете в открытом доступе Telco Customer Churn (Набор данных о клиентах телекоммуникационной компании). Инструментом анализа и создания моделей предсказания будут использованы язык Python в среде Jupyter Notebook.

Актуальность работы заключается в том, что по сей день существует огромное количество компаний и бизнесов, для которых жизненно важно знать об оттоке клиентов. Поэтому данная работа должна внести ясность в эту тему.

Новизна заключается в том, что вместо классической веб-оболочки Jupyter notebook было отдано предпочтение Google Collab. Colaboratory получает удаленный доступ к бесплатной машине с видеокартой, что сильно упрощает работу, особенно в контексте создания моделей машинного обучения. Помимо того, Google Collab легко справляется с установкой пакетов и фреймворков, что не всегда наблюдается при работе локально в счёт особенности установок у каждого компьютера.

2. Основные понятия в задачах предсказания оттока клиентов.

Компании обычно проводят различие между добровольным оттоком и непроизвольным оттоком. Добровольный отток происходит из-за решения клиента переключиться на другую компанию или поставщика услуг, непроизвольный отток происходит из-за таких обстоятельств, как переезд клиента в учреждение долгосрочного ухода, смерть или переезд в отдаленное место. В большинстве случаев непроизвольные причины оттока исключаются из аналитических моделей. Аналитики склонны концентрироваться на добровольном оттоке, поскольку он обычно происходит из-за факторов взаимоотношений компании и клиента, которые контролируются самими компании, таких как то, как обрабатываются взаимодействия по выставлению счетов или как предоставляется послепродажная помощь.

Когда компании измеряют оборот своих клиентов, они обычно проводят различие между валовым и чистым истощением. Валовое истощение-это потеря существующих клиентов и связанных с ними периодических доходов от контрактных товаров или услуг в течение определенного периода. Чистое истощение-это валовое истощение плюс добавление или набор аналогичных клиентов в первоначальном местоположении. Финансовые учреждения часто отслеживают и измеряют потери, используя взвешенный расчет, называемый ежемесячным повторяющимся доходом (или MRR). В 2000-х годах появилось также несколько программ бизнес-аналитики, которые могут обрабатывать базы данных информации о клиентах и анализировать факторы, связанные с истощением клиентов, такие как неудовлетворенность обслуживанием или технической поддержкой, споры о

выставлении счетов или разногласия по поводу политики компании. Более сложные прогностические аналитические программы используют модели прогнозирования оттока, которые предсказывают отток клиентов, оценивая их склонность к риску оттока. Поскольку эти модели генерируют небольшой список потенциальных перебежчиков с приоритетами, они эффективно фокусируют маркетинговые программы удержания клиентов на той части клиентской базы, которая наиболее уязвима для оттока.

В нашем случае задача будет состоять в классификации клиентов как потенциально отказывающегося от наших услуг и тех, кто не собирается уходить.

3. Описание выбранных данных.

Каждая строка представляет клиента, каждый столбец содержит атрибуты клиента. Набор данных включает в себя информацию о 7043 клиентах про:

Клиенты, которые ушли в течение последнего месяца – колонка называется churn

Услуги, на которые подписался каждый клиент – телефон, несколько линий, интернет, онлайн-безопасность, онлайн-резервное копирование, защита устройств, техническая поддержка, а также потоковое телевидение и фильмы

Информация о счете клиента – как долго он был клиентом, контракт, способ оплаты, безбумажный биллинг, ежемесячные платежи и общие сборы

Демографическая информация о клиентах – пол, возрастной диапазон, а также есть ли у них партнеры и иждивенцы.

Описание каждого столбца подробно:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No

Figure 1 Первоначальный вид датафрейма 1

DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

Figure 2 Первоначальный вид датафрейма 2

CustomerID: Идентификационный номер клиента.

Gender: Является ли клиент мужчиной или женщиной.

SeniorCitizen: Является ли клиент пожилым гражданином или нет (1, 0)

Partner: Есть ли у клиента партнер или нет (да, нет)

Dependents: Есть ли у клиента иждивенцы или нет (да, нет)

Tenure: Количество месяцев, в течение которых клиент находился в компании

PhoneService: Находится ли клиент в телефонном обслуживании (да, нет)

MultipleLines: Имеет ли у клиента несколько линий ссвязи

InternetService: Интернет-провайдер клиента (DSL, волоконно-оптический, нет)

OnlineSecurity: Есть ли у клиента онлайн-безопасность или нет (Да, Нет, нет интернет-сервиса)

OnlineBackup: Есть ли у клиента онлайн-резервная копия или нет (Да, Нет, нет интернет-сервиса)

DeviceProtection: Есть ли у клиента защита устройства или нет (Да, Нет, нет интернет-сервиса)

TechSupport: Есть ли у клиента техническая поддержка или нет (Да, Нет, нет интернет-сервиса)

StreamingTV: Есть ли у клиента потоковое телевидение или нет (Да, Нет, нет интернет-сервиса)

StreamingMovies: Есть ли у клиента потоковые фильмы или нет (Да, Нет, нет интернет-сервиса)

Contract: Срок действия договора заказчика (от месяца до месяца, один год, два года)

PaperlessBilling: Независимо от того, есть ли у клиента безбумажный биллинг или нет (да, нет)

PaymentMethod: Способ оплаты клиента (электронный чек, почтовый чек, банковский перевод (автоматический), кредитная карта (автоматический))

MonthlyCharges: Сумма, взимаемая с клиента ежемесячно

TotalCharges: Общая сумма, начисленная клиентом

Churn: Сам маркер оттока (относится ли к группе клиентов, которая планирует отказаться от услуг)

4. Предварительный анализ и обработка данных.

Загрузка данных в датафрейм при помощи библиотеки pandas

```
df = pd.read_csv('drive/MyDrive/Colab Notebooks/CustomerChurn.csv')
df.head(5)
```

Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	De
No	1	No	No phone service	DSL	No	Yes	
No	34	Yes	No	DSL	Yes	No	
No	2	Yes	No	DSL	Yes	Yes	
No	45	No	No phone service	DSL	Yes	No	
No	2	Yes	No	Fiber optic	No	No	

Figure 3 Инициализация датафрейма

Если взглянуть на данные, можно понять, что нужно исключить столбец, отвечающий за идентификационный номер клиента. Далее проверка типов данных выбранного датасета.

```
df.drop('customerID',axis='columns',inplace=True)
df.dtypes
```

```
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges        object
Churn               object
dtype: object
```

Figure 4 Описание данных по типам

Видно, что столбец TotalCharges, отвечающий за общую сумму, которую выплатил клиент, содержит данные типа object, а должен float. Проверка.

```
df.TotalCharges.values
```

```
array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
      dtype=object)
```

Figure 5 Проверка типа данных в столбце TotalCharges

Видно, что данные в этом столбце представлены строками. Смена типа на числовой.

```
pd.to_numeric(df.TotalCharges)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
pandas/_libs/lib.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " "

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-8-06ba430a4ba5> in <module>()
----> 1 pd.to_numeric(df.TotalCharges)

/usr/local/lib/python3.6/dist-packages/pandas/core/tools/numeric.py in to_numeric(arg, errors, downcast)
    151         try:
    152             values = lib.maybe_convert_numeric(
--> 153                 values, set(), coerce_numeric=coerce_numeric
    154             )
    155         except (ValueError, TypeError):

pandas/_libs/lib.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " " at position 488
```

Figure 6 Ошибка смены типа TotalCharges на числовой

Видно ошибку, в 488ой строке стоит пропуск. Тут избавление от всех строк, в которых TotalCharges не содержит значения. Пропуск не значит, что клиент ещё не заплатил. В таком случае стоял бы 0. Поэтому, чтобы не навредить моделям в дальнейшем, избавляются от таких записей.

```
[14] df1 = df[df.TotalCharges!=' ']
     df1.shape

     (7032, 20)
```

Figure 7 Удаление записей с пробелами

Удалено 11 записей.

Визуализация оттока от времени, проведенного клиентом в сотрудничестве, и от ежемесячной платы.

По времени

```
[19] tenure_churn_no = df1[df1.Churn=='No'].tenure
     tenure_churn_yes = df1[df1.Churn=='Yes'].tenure
     plt.xlabel("tenure")
     plt.ylabel("Number Of Customers")
     plt.title("Customer Churn Prediction Visualiztion")
     plt.hist([tenure_churn_yes, tenure_churn_no], rwidth=0.95, color=['green','red'],label=['Churn=Yes','Churn=No'])
     plt.legend()
```

Figure 8 Код для гистограммы по времени

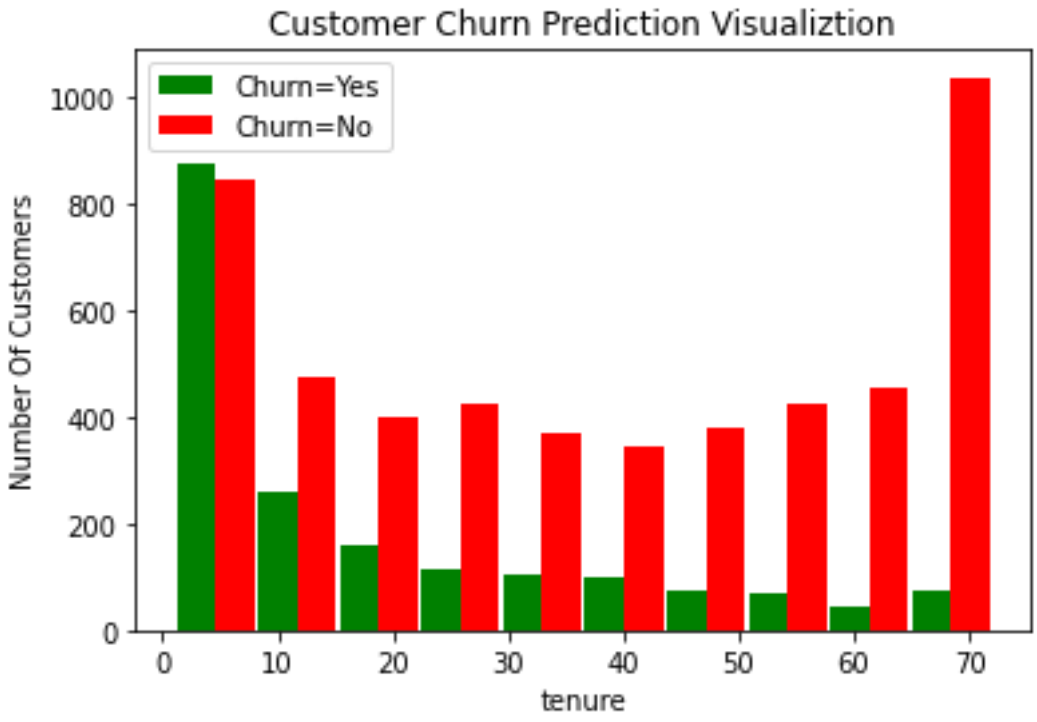

Figure 9 Гистограмма зависимости оттока количества пользователей по времени

Из графика видно, что в первые месяцы пользования, их количество не влияет на фактор ухода. Что касается всех остальных промежутков, чем дольше клиент им является, тем меньше шанс того, что он перестанет им быть.

По ежемесячной оплате.

```
[20] mc_churn_no = df1[df1.Churn=='No'].MonthlyCharges
     mc_churn_yes = df1[df1.Churn=='Yes'].MonthlyCharges
     plt.xlabel("Monthly Charges")
     plt.ylabel("Number Of Customers")
     plt.title("Customer Churn Prediction Visualiztion")
     plt.hist([mc_churn_yes, mc_churn_no], rwidth=0.95, color=['green','red'],label=['Churn=Yes','Churn=No'])
     plt.legend()
```

Figure 10 Код для гистограммы по месячной плате

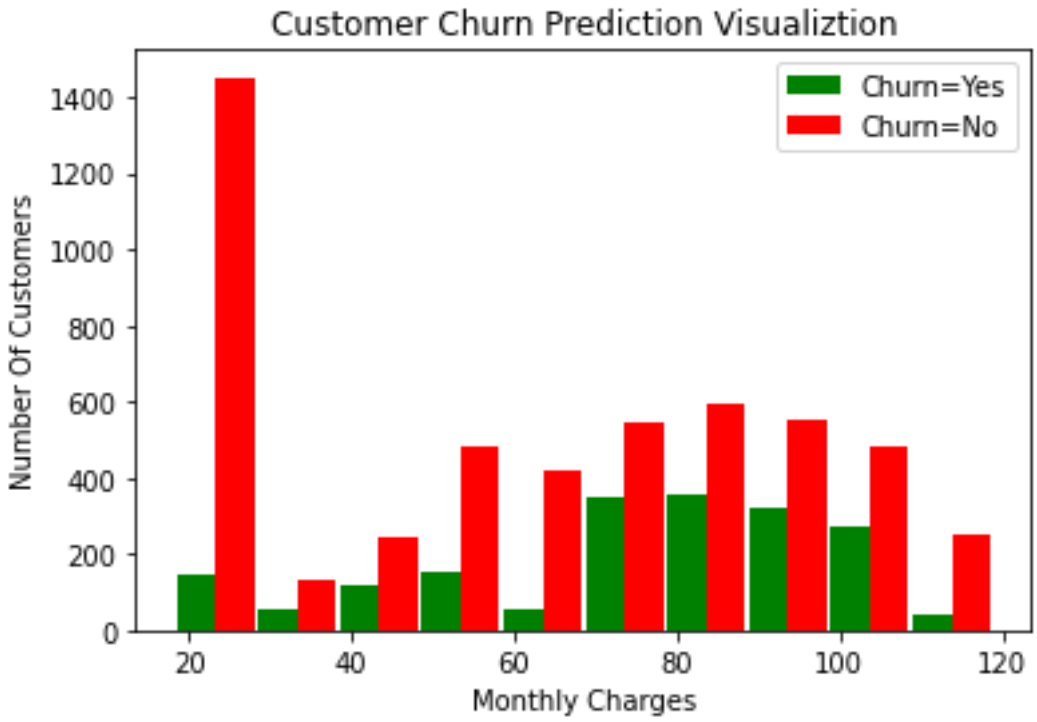

Figure 11 Гистограмма зависимости оттока количества пользователей по деньгам, оплачиваемым ими в месяц

Из графика видно, что при минимальных затратах, клиент чувствует себя комфортно. При больших затратах, клиенту трудно так просто перейти от сервиса, т.к. либо уже много вложил денег, либо пользуется большим спектром услуг. На среднем промежутке же, клиенту предоставляется пространство для размышлений о том, выгодно ему это, или нет.

Стоит заметить, что присутствуют много категориальных переменных. Для удобства в их обработке, нужно преобразовать переменные в числа.

Проверка всех значения.

```
[21] def print_unique_col_values(df):
           for column in df:
                if df[column].dtypes=='object':
                    print(f'{column}: {df[column].unique()}')
     print_unique_col_values(df1)
```

```
gender: ['Female' 'Male']
Partner: ['Yes' 'No']
Dependents: ['No' 'Yes']
PhoneService: ['No' 'Yes']
MultipleLines: ['No phone service' 'No' 'Yes']
InternetService: ['DSL' 'Fiber optic' 'No']
OnlineSecurity: ['No' 'Yes' 'No internet service']
OnlineBackup: ['Yes' 'No' 'No internet service']
DeviceProtection: ['No' 'Yes' 'No internet service']
TechSupport: ['No' 'Yes' 'No internet service']
StreamingTV: ['No' 'Yes' 'No internet service']
StreamingMovies: ['No' 'Yes' 'No internet service']
Contract: ['Month-to-month' 'One year' 'Two year']
PaperlessBilling: ['Yes' 'No']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

Figure 12 Проверка уникальных значений

Для некоторых переменных No и No internet service – одно и то же. Замена их на No.

```
[22] df1.replace('No internet service','No',inplace=True)
     df1.replace('No phone service','No',inplace=True)
```

Figure 13 Наличие сторонних услуг код

Преобразование в числа.

```
[24] yes_no_columns = ['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup',
                       'DeviceProtection','TechSupport','StreamingTV','StreamingMovies','PaperlessBilling','Churn']
     for col in yes_no_columns:
         df1[col].replace({'Yes': 1,'No': 0},inplace=True)
```

Figure 14 Преобразование в числа категориальные переменные код

И для столбца, отвечающего за пол.

```
[25] df1['gender'].replace({'Male': 1, 'Female': 0}, inplace = True)
```

Figure 15 Изменение столбца пол

А так же для столбцов, имеющих больше двух категорий, деление на отдельные.

```
df2 = pd.get_dummies(data=df1, columns=['InternetService','Contract','PaymentMethod'])
df2.columns
```

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'InternetService_DSL', 'InternetService_Fiber optic',
       'InternetService_No', 'Contract_Month-to-month', 'Contract_One year',
       'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)',
       'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check'],
      dtype='object')
```

Figure 16 Получение из категориальных переменных бинарные столбцы

Теперь датафрейм состоит только из чисел. Часть столбцов:

```
[28] df2.sample(5)
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup
299	0	0	0	1	1	1	0	1	0
4615	0	0	1	0	72	1	1	1	1
5681	0	1	1	0	34	1	0	0	0
3386	0	0	1	0	58	1	1	1	0
3883	0	0	0	0	12	1	1	1	0

Figure 17 Взгляд на обработанный датафрейм

Далее нужно проскалировать числовые переменные для ускорения вычислений.

```
[30] cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']

     scaler = MinMaxScaler()
     df2[cols_to_scale] = scaler.fit_transform(df2[cols_to_scale])
```

```
[31] for col in df2:
         print(f'{col}: {df2[col].unique()}')
```

```
gender: [0 1]
SeniorCitizen: [0 1]
Partner: [1 0]
Dependents: [0 1]
tenure: [0.         0.46478873 0.01408451 0.61971831 0.09859155 0.29577465
 0.12676056 0.38028169 0.85915493 0.16901408 0.21126761 0.8028169
 0.67605634 0.33802817 0.95774648 0.71830986 0.98591549 0.28169014
 0.15492958 0.4084507  0.64788732 1.         0.22535211 0.36619718
 0.05633803 0.63380282 0.14084507 0.97183099 0.87323944 0.5915493
 0.1971831  0.83098592 0.23943662 0.91549296 0.11267606 0.02816901
 0.42253521 0.69014085 0.88732394 0.77464789 0.08450704 0.57746479
 0.47887324 0.66197183 0.3943662  0.90140845 0.52112676 0.94366197
 0.43661972 0.76056338 0.50704225 0.49295775 0.56338028 0.07042254
 0.04225352 0.45070423 0.92957746 0.30985915 0.78873239 0.84507042
 0.18309859 0.26760563 0.73239437 0.54929577 0.81690141 0.32394366
 0.6056338  0.25352113 0.74647887 0.70422535 0.35211268 0.53521127]
PhoneService: [0 1]
MultipleLines: [0 1]
OnlineSecurity: [0 1]
OnlineBackup: [1 0]
DeviceProtection: [0 1]
TechSupport: [0 1]
StreamingTV: [0 1]
StreamingMovies: [0 1]
PaperlessBilling: [1 0]
MonthlyCharges: [0.11542289 0.38507463 0.35422886 ... 0.44626866 0.25820896 0.60149254]
TotalCharges: [0.0012751  0.21586661 0.01031041 ... 0.03780868 0.03321025 0.78764136]
Churn: [0 1]
InternetService_DSL: [1 0]
InternetService_Fiber optic: [0 1]
InternetService_No: [0 1]
Contract_Month-to-month: [1 0]
Contract_One year: [0 1]
Contract_Two year: [0 1]
PaymentMethod_Bank transfer (automatic): [0 1]
PaymentMethod_Credit card (automatic): [0 1]
PaymentMethod_Electronic check: [1 0]
PaymentMethod_Mailed check: [0 1]
```

Figure 18 Скалирование

Теперь данные готовы к обработке.

5. Описание моделей и их имплементация

5.1 Выбранные модели

Для предиктивных моделей были выбраны алгоритмы: Наивный Баес, Стохастический Градиентный Спуск (классификатор), Классификатор К-ближайших соседей, Дерево Решений, Случайный Лес, Машина Опорных Векторов, Логистическая регрессия, XGBoost классификатор.

Перейдём к описанию каждого из них

5.2 Наивный Баес

Наивные байесовские классификаторы[1] построены на байесовских методах классификации. Они опираются на теорему Байеса, которая представляет собой уравнение, описывающее соотношение условных вероятностей статистических величин. В Байесовской классификации нужно быть заинтересованным в том, чтобы найти вероятность метки с учетом некоторых наблюдаемых признаков, которые можно записать как

$$P(L \mid \text{features})$$

Теорема Байеса говорит, как выразить это в терминах величин, которые можно вычислить более непосредственно:

$$P(L \mid \text{features}) = \frac{P(\text{features} \mid L)P(L)}{P(\text{features})}$$

Метод зовётся наивным в счёт того, что выдвигается предположение о независимости признаков.

5.3 Стохастический Градиентный Спуск

Стохастический Градиентный Спуск – один из методов классификации градиентным спуском, который заключается в минимизации функции ошибок путём спуска градиентом.

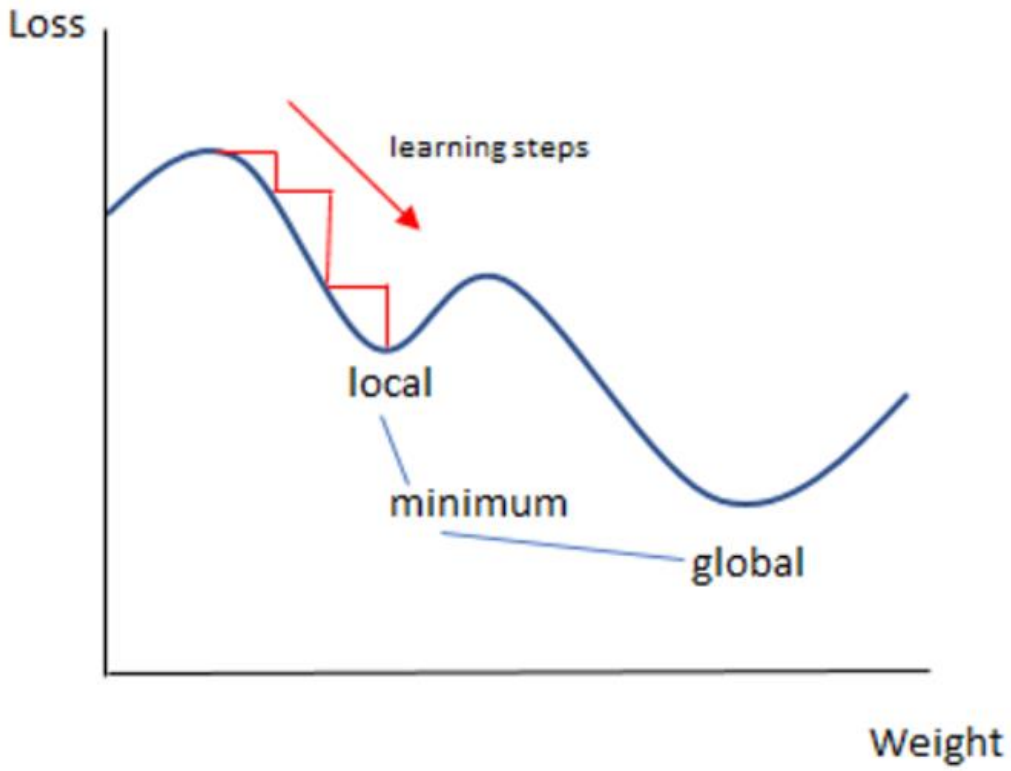

Figure 19 Схема работы SGDB

Но в отличие от обычного градиентного спуска, SGD спускается градиентом, вычисленным не по всему набору данных, а по случайной выборке.

$$w_{t+1} = w_t - \eta_t(\lambda w_t + \nabla \ell(w_t, x_t, y_t))$$

Figure 20 вычисление градиента

5.4 К-ближайших соседей

Обучающие примеры представляют собой векторы в многомерном пространстве объектов, каждый из которых имеет метку класса. Обучающая фаза алгоритма состоит только из хранения векторов признаков и меток классов обучающих выборок.

На этапе классификации k является определяемой пользователем константой, а немеченый вектор (запрос или тестовая точка) классифицируется путем присвоения метки, которая наиболее часто встречается среди k обучающих выборок, ближайших к этой точке запроса.

Обычно используемой метрикой расстояния для непрерывных переменных является Евклидово расстояние. Для дискретных переменных, например для классификации текста, можно использовать другую метрику, например метрику перекрытия (или расстояние Хэмминга).

Например,

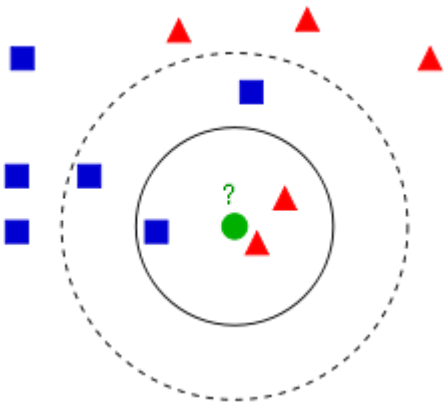

Figure 21 Классификация с помощью KNN

Пример классификации KNN. Тестовый образец (зеленая точка) должен быть классифицирован либо на синие квадраты, либо на красные треугольники. Если k = 3 (круг сплошной линии), то он присваивается красным треугольникам, потому что внутри внутреннего круга есть 2 треугольника и только 1 квадрат. Если k = 5 (пунктирный круг), то он присваивается синим квадратам (3 квадрата против 2 треугольников внутри внешнего круга).

5.5 Дерево Решений

Дерево решений - это простое представление для классификации примеров. Для этого предположим, что все входные объекты имеют конечные дискретные области, и существует один целевой объект, называемый "классификацией". Каждый элемент предметной области классификации называется классом. Дерево решений или дерево классификации-это дерево, в котором каждый внутренний (не листовой) узел помечен входным объектом. Дуги, поступающие из узла, помеченного входным объектом, помечаются каждым из возможных значений целевого объекта, или дуга ведет к

подчиненному узлу принятия решения по другому входному объекту. Каждый лист дерева помечен классом или распределением вероятностей по классам, что означает, что набор данных был классифицирован деревом либо в определенный класс, либо в определенное распределение вероятностей (которое, если дерево решений хорошо построено, смещено в сторону определенных подмножеств классов).

Дерево строится путем разбиения исходного набора, составляющего корневой узел дерева, на подмножества, которые образуют дочерние элементы-преемники. Расщепление основано на наборе правил расщепления, основанных на классификационных признаках. Этот процесс повторяется на каждом производном подмножестве рекурсивным способом, называемым рекурсивным разбиением. Рекурсия завершается, когда подмножество в узле имеет все те же значения целевой переменной или когда расщепление больше не добавляет значения прогнозам. Этот процесс нисходящей индукции деревьев решений является примером жадного алгоритма и на сегодняшний день является наиболее распространенной стратегией изучения деревьев решений по данным.

5.6 Случайный Лес

Случайный лес, как и следует из его названия, состоит из большого числа отдельных деревьев решений, которые действуют как ансамбль. Каждое отдельное дерево в случайном лесу ‘выплевывает’ предсказание класса, и класс с наибольшим количеством голосов становится предсказанием нашей модели.

Фундаментальная концепция, лежащая в основе случайного леса, проста, но сильна — мудрость толпы. В науке о данных говорят, что причина того, что модель случайного леса работает так хорошо, заключается в том, что:

Большое число относительно некоррелированных моделей (деревьев), действующих в качестве комитета, будет превосходить любую из отдельных составляющих моделей.

Ключевым фактором является низкая корреляция между моделями. Точно так же, как инвестиции с низкой корреляцией (например, акции и облигации) объединяются вместе, чтобы сформировать портфель, который больше, чем сумма его частей, некоррелированные модели могут производить ансамблевые предсказания, которые более точны, чем любые индивидуальные предсказания. Причина этого удивительного эффекта заключается в том, что деревья защищают друг друга от своих индивидуальных

ошибок (до тех пор, пока они не будут постоянно ошибаться в одном и том же направлении). В то время как некоторые деревья могут быть неправильными, многие другие деревья будут правильными, так что как группа деревьев способна двигаться в правильном направлении. Таким образом, предпосылками для успешной работы случайного леса являются:

5.7 Машина Опорных Векторов

Задача алгоритма машины опорных векторов [2] состоит в том, чтобы найти гиперплоскость в N-мерном пространстве(N — число признаков), которая отчетливо классифицирует точки данных.

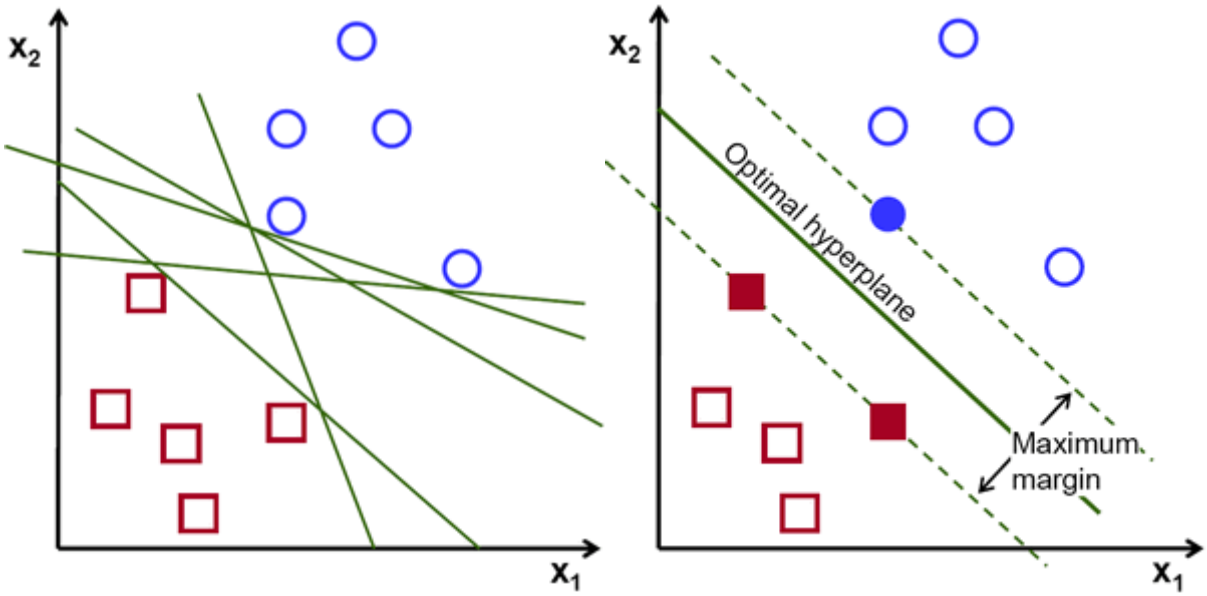

Figure 22 Пример оптимальных гиперплоскостей

Figure 23 Примерные гиперплоскости

Для разделения двух классов точек данных можно выбрать множество возможных гиперплоскостей. Наша цель состоит в том, чтобы найти плоскость, которая имеет максимальный запас, то есть максимальное расстояние между точками данных обоих классов. Максимизация расстояния маржи обеспечивает некоторое подкрепление, так что будущие точки данных могут быть классифицированы с большей уверенностью.

Гиперплоскости-это границы принятия решений, которые помогают классифицировать точки данных. Точки данных, падающие по обе стороны гиперплоскости, можно отнести к разным классам. Кроме того, размерность гиперплоскости зависит от количества объектов. Если число входных объектов равно 2, то гиперплоскость-это просто линия.

Если число входных объектов равно 3, то гиперплоскость становится двумерной плоскостью. Это становится трудно представить, когда количество функций превышает 3.

Опорные векторы-это точки данных, которые находятся ближе к гиперплоскости и влияют на положение и ориентацию гиперплоскости. Используя эти опорные векторы, мы максимизируем маржу классификатора. Удаление опорных векторов изменит положение гиперплоскости. Это те моменты, которые помогают нам построить нашу SVM.

5.8 Логистическая регрессия

Логистическая регрессия является одной из линейных моделей классификации. В этой модели вероятности описывают исходы одного испытания и моделируются с помощью логистической функции. В случае модели с двумя предикторами для бинарной классификации можно описать модель как

$$\ell = \log_b\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

Тогда вероятность $Y = 1$:

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1}$$

5.9 XGBoost

XGBoost-это популярная реализация градиентного бустинга..

Регуляризация: XGBoost имеет возможность штрафовать сложные модели как с помощью регуляризации L1, так и с помощью регуляризации L2. Регуляризация помогает предотвратить переобучение

Обработка разреженных данных: пропущенные значения или этапы обработки данных, такие как однократное кодирование, делают данные разреженными. XGBoost включает в себя алгоритм поиска расщепления с учетом разреженности для обработки различных типов паттернов разреженности в данных

Взвешенный квантильный эскиз: большинство существующих алгоритмов на основе дерева могут найти точки разделения, когда точки данных имеют равные веса (используя алгоритм квантильного эскиза). Однако они не оборудованы для обработки взвешенных данных. XGBoost имеет распределенный взвешенный алгоритм квантильного эскиза для эффективной обработки взвешенных данных

Блочная структура для параллельного обучения: для более быстрых вычислений XGBoost может использовать несколько ядер на процессоре. Это возможно из - за блочной структуры в его системном дизайне. Данные сортируются и хранятся в блоках памяти, называемых блоками. В отличие от других алгоритмов, это позволяет повторно использовать макет данных в последующих итерациях, а не вычислять его снова. Эта функция также полезна для таких шагов, как поиск разбиения и суб-выборка столбцов

Осведомленность о кэше: в XGBoost для получения статистики градиента по индексу строк требуется непрерывный доступ к памяти. Таким образом, XGBoost был разработан для оптимального использования аппаратного обеспечения. Это делается путем выделения внутренних буферов в каждом потоке, где может храниться статистика градиента

Внеядерные вычисления: эта функция оптимизирует доступное дисковое пространство и максимизирует его использование при обработке огромных наборов данных, которые не помещаются в память

5.10 Имплементация

Инициализация эндогенных и экзогенных переменных.

```
[32] X = df2.drop('Churn',axis='columns')
     y = df2['Churn']
```

Figure 24 Инициализация X и Y

Создание набора пустых моделей с заданными параметрами

```
[33] models = [GaussianNB(),
               SGDClassifier(max_iter=5000, random_state=0),
               KNeighborsClassifier(n_neighbors=19),
               DecisionTreeClassifier(),
              RandomForestClassifier(n_estimators=1000,
                                     max_depth=10, random_state=0),
              SVC(decision_function_shape="ovo"),
               LogisticRegression(random_state=0,
                                  solver='lbfgs', multi_class='multinomial'),
              XGBClassifier(n_estimators=1000,
                            learning_rate=0.05)]
     model_names = ['Naive Bayes',
                    'Stochastic Gradient Descent',
                    'KNN', 'Decision trees',
                    'Random Forest', 'Support Vector Machine',
                    'Logistic Regression',
                    'Cross Gradient Booster']
```

Figure 25 Инициализация моделей

Нужно создать пайплайн который на вход получает полную выборку признаков и целевых переменных, а на выходе массив. В этом массиве находится наилучшая обученная на входных данных модель, выбранная по принципу точности, её имя и само значение точности

```
[34] %time
     def pipeline(X, y):
         mx = 0
         mxmdl = 0
         X_train, X_test, y_train, y_test = train_test_split(X,
                                                             y,
                                                             test_size=0.3,
                                                             random_state=42)
         for i, model in enumerate(models):
             %time
             model.fit(X_train, y_train)
             a = accuracy_score(y_test, model.predict(X_test))
             if a > mx:
                 mx = a
                 mxmdl = model
                 mxmdl_name = model_names[i]
             print(model_names[i], a)
         print(mxmdl_name, ' has the best accuracy of ', mx)
         return(mxmdl, mxmdl_name, mx)
     fmodel = pipeline(X, y)[0]
     fmodel
```

```
CPU times: user 3 µs, sys: 1 µs, total: 4 µs
Wall time: 7.39 µs
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 5.01 µs
Naive Bayes 0.7388625592417062
CPU times: user 3 µs, sys: 1 µs, total: 4 µs
Wall time: 5.96 µs
Stochastic Gradient Descent 0.781042654028436
CPU times: user 6 µs, sys: 1 µs, total: 7 µs
Wall time: 7.15 µs
KNN 0.7739336492890996
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.01 µs
Decision trees 0.7293838862559242
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 5.48 µs
Random Forest 0.795260663507109
CPU times: user 2 µs, sys: 0 ns, total: 2 µs
Wall time: 4.77 µs
Support Vector Machine 0.7933649289099526
CPU times: user 3 µs, sys: 1e+03 ns, total: 4 µs
Wall time: 5.25 µs
Logistic Regression 0.795260663507109
CPU times: user 7 µs, sys: 1e+03 ns, total: 8 µs
Wall time: 7.15 µs
Cross Gradient Booster 0.7914691943127962
Random Forest  has the best accuracy of  0.795260663507109
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Figure 26 Пайплайн

В результате получилась бученную модель Случайного Леса с точностью ~ 0.7952. Попытаемся улучшить модель.

5.11 Подбор Гиперпараметров

Часто общие эффекты гиперпараметров на модель известны, но как лучше всего установить гиперпараметр и комбинации взаимодействующих гиперпараметров для данного набора данных, является сложной задачей. Часто существуют общие эвристики или эмпирические правила для настройки гиперпараметров.[3]

Лучший подход заключается в объективном поиске различных значений гиперпараметров модели и выборе подмножества, которое приводит к модели, достигающей наилучшей производительности в данном наборе данных. Это называется оптимизацией гиперпараметров или настройкой гиперпараметров и доступно в библиотеке машинного обучения scikit-learn Python. Результатом оптимизации гиперпараметров является единый набор хорошо работающих гиперпараметров, который можно использовать для настройки модели.

Гиперпараметрическая оптимизация необходима, чтобы получить максимальную отдачу от моделей машинного обучения.

Можно использовать целый ряд различных алгоритмов оптимизации, хотя два самых простых и распространенных метода-это случайный поиск и поиск по сетке.

RandomSearch. Определите пространство поиска как ограниченную область значений гиперпараметров и произвольно выбранных точек в этой области.

GridSearch. Определите пространство поиска в виде сетки значений гиперпараметров и оцените каждую позицию в этой сетке.

```
[36] X_train, X_test, y_train, y_test = train_test_split(X,
                                                         y,
                                                         test_size=0.3,
                                                         random_state=42)
     gs = GridSearchCV(estimator = fmodel,
                       param_grid = {'n_estimators': np.arange(900, 1100, 50), 'criterion': ['gini', 'entropy']},
                      cv = 4, verbose = 2)
     br = gs.fit(X_train, y_train)

     accuracy_score(y_test, br.predict(X_test))
```

Figure 27 Поиск по сетке

```
[36] [CV] criterion=gini, n_estimators=900 ...............................
     [CV] ................. criterion=gini, n_estimators=900, total=   3.9s
     [CV] criterion=gini, n_estimators=950 ...............................
     [CV] ................. criterion=gini, n_estimators=950, total=   4.2s
     [CV] criterion=gini, n_estimators=950 ...............................
     [CV] ................. criterion=gini, n_estimators=950, total=   4.4s
     [CV] criterion=gini, n_estimators=950 ...............................
     [CV] ................. criterion=gini, n_estimators=950, total=   4.3s
     [CV] criterion=gini, n_estimators=950 ...............................
     [CV] ................. criterion=gini, n_estimators=950, total=   4.5s
     [CV] criterion=gini, n_estimators=1000 ..............................
     [CV] ................. criterion=gini, n_estimators=1000, total=   4.4s
     [CV] criterion=gini, n_estimators=1000 ..............................
     [CV] ................. criterion=gini, n_estimators=1000, total=   4.5s
     [CV] criterion=gini, n_estimators=1000 ..............................
     [CV] ................. criterion=gini, n_estimators=1000, total=   4.7s
     [CV] criterion=gini, n_estimators=1000 ..............................
     [CV] ................. criterion=gini, n_estimators=1000, total=   4.3s
     [CV] criterion=gini, n_estimators=1050 ..............................
     [CV] ................. criterion=gini, n_estimators=1050, total=   4.6s
     [CV] criterion=gini, n_estimators=1050 ..............................
     [CV] ................. criterion=gini, n_estimators=1050, total=   4.7s
     [CV] criterion=gini, n_estimators=1050 ..............................
     [CV] ................. criterion=gini, n_estimators=1050, total=   4.6s
     [CV] criterion=gini, n_estimators=1050 ..............................
     [CV] ................. criterion=gini, n_estimators=1050, total=   4.8s
     [CV] criterion=entropy, n_estimators=900 ............................
     [CV] ............... criterion=entropy, n_estimators=900, total=   5.2s
     [CV] criterion=entropy, n_estimators=900 ............................
     [CV] ............... criterion=entropy, n_estimators=900, total=   4.8s
     [CV] criterion=entropy, n_estimators=900 ............................
     [CV] ............... criterion=entropy, n_estimators=900, total=   5.0s
     [CV] criterion=entropy, n_estimators=900 ............................
     [CV] ............... criterion=entropy, n_estimators=900, total=   5.3s
     [CV] criterion=entropy, n_estimators=950 ............................
     [CV] ............... criterion=entropy, n_estimators=950, total=   4.9s
     [CV] criterion=entropy, n_estimators=950 ............................
     [CV] ............... criterion=entropy, n_estimators=950, total=   5.2s
     [CV] criterion=entropy, n_estimators=950 ............................
     [CV] ............... criterion=entropy, n_estimators=950, total=   5.4s
     [CV] criterion=entropy, n_estimators=950 ............................
     [CV] ............... criterion=entropy, n_estimators=950, total=   4.9s
     [CV] criterion=entropy, n_estimators=1000 ...........................
     [CV] ............... criterion=entropy, n_estimators=1000, total=   5.3s
     [CV] criterion=entropy, n_estimators=1000 ...........................
     [CV] ............... criterion=entropy, n_estimators=1000, total=   5.4s
     [CV] criterion=entropy, n_estimators=1000 ...........................
     [CV] ............... criterion=entropy, n_estimators=1000, total=   5.5s
     [CV] criterion=entropy, n_estimators=1000 ...........................
     [CV] ............... criterion=entropy, n_estimators=1000, total=   5.3s
     [CV] criterion=entropy, n_estimators=1050 ...........................
     [CV] ............... criterion=entropy, n_estimators=1050, total=   5.9s
     [CV] criterion=entropy, n_estimators=1050 ...........................
     [CV] ............... criterion=entropy, n_estimators=1050, total=   5.4s
     [CV] criterion=entropy, n_estimators=1050 ...........................
     [CV] ............... criterion=entropy, n_estimators=1050, total=   5.8s
     [CV] criterion=entropy, n_estimators=1050 ...........................
     [CV] ............... criterion=entropy, n_estimators=1050, total=   5.8s
     [Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:  2.6min finished
     0.7976303317535545
```

Figure 28 Обучение

Добиться лучшей точности не получилось при помощи GridSearch. Принята попытка исследовать с помощью RandomSearch. Задание сетки параметров.

```
[37] n_estimators = np.arange(900, 1100, 25)
     max_features = ['auto', 'sqrt']
     max_depth = np.arange(10, 120, 10)
     np.append(max_depth, None)
     bootstrap = [True, False]


     random_grid = {'n_estimators': n_estimators,
                    'max_features': max_features,
                    'max_depth': max_depth,
                    'bootstrap': bootstrap}
```

Figure 29 Сетка параметров для случайного поиска

```
[38] rf_random = RandomizedSearchCV(estimator = fmodel,
                                    param_distributions = random_grid,
                                    n_iter = 100,
                                    cv = 3,
                                    verbose=2,
                                    random_state=42,
                                    n_jobs = -1)
     rf_random.fit(X_train, y_train)
     accuracy_score(y_test, rf_random.predict(X_test))

     Fitting 3 folds for each of 100 candidates, totalling 300 fits
     [Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
     [Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:  2.5min
     [Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed: 10.2min
     [Parallel(n_jobs=-1)]: Done 300 out of 300 | elapsed: 19.8min finished
     0.795260663507109
```

Figure 30 Поиск случайный

За 20 минут вычислений, подобрать параметры лучше, не получилось. Сохранение модели в финальную версию.

6. Заключение

В итоге нужно посмотреть на то, как выглядит распределение классов.

```
[44] preds = fmodel.predict(X)
     print('Точность', ':', round(accuracy_score(y, preds), 5), '\n')

     # Матрица сравнений
     confusion_matr = confusion_matrix(y, preds, normalize='all') #normalize = 'true'
     plt.figure(figsize = (16, 9))
     sns.heatmap(confusion_matr, annot=True, xticklabels= ['НО', 'ВП'], yticklabels=['ВО', 'НП'])
     plt.savefig("conf matrix")

     Точность : 0.85609
```

Figure 31 Код для матрицы классов

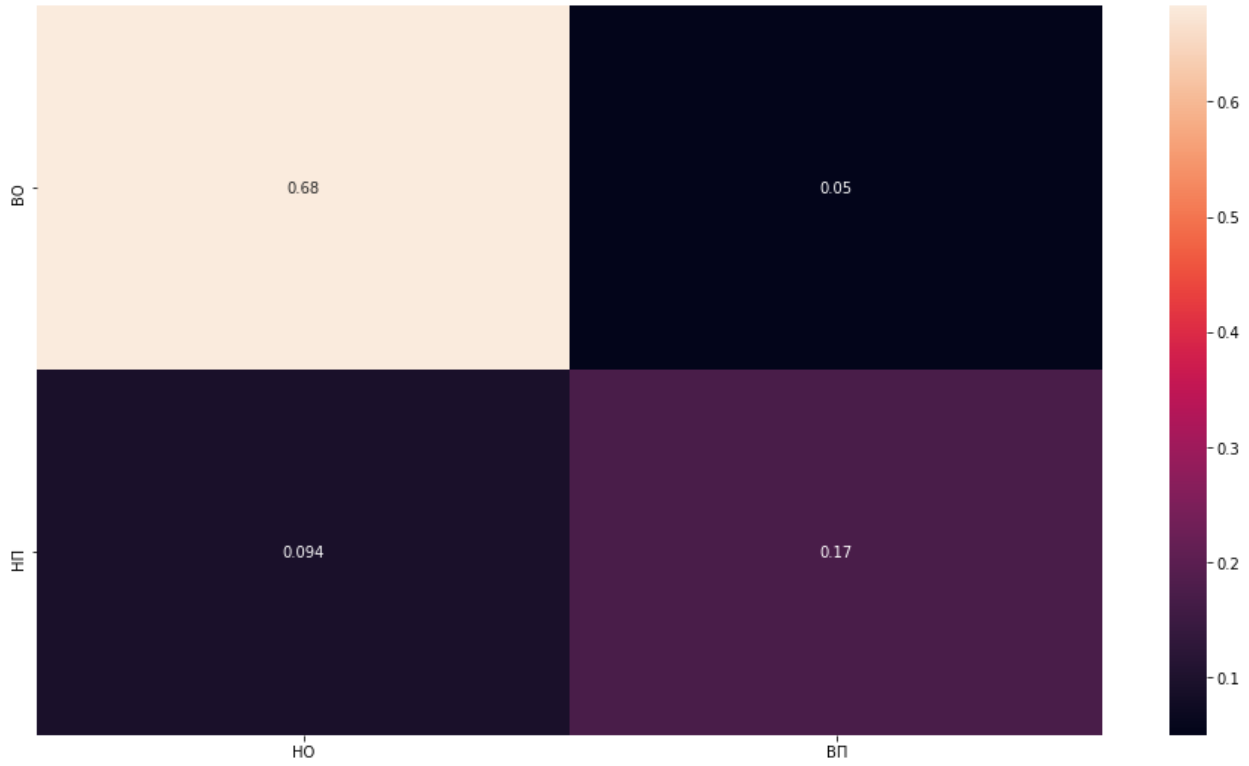

Figure 32 Матрица классов

Видно распределение классов В/Н = Верно/Неверно, П/О = Положительный/Отрицательный

Стоит посмотреть на весы значимости признаков

```
[42] #Посмотрим на отдельные весы значимости признаков
     perm = PermutationImportance(estimator=fmodel, random_state=1)
     perm.fit(X_test, y_test)
     eli5.show_weights(estimator=perm, feature_names = X_test.columns.tolist())
```

Figure 33 Таблица весов код

Weight	Feature
0.0236 ± 0.0047	tenure
0.0144 ± 0.0081	InternetService_Fiber optic
0.0129 ± 0.0043	Contract_Month-to-month
0.0100 ± 0.0025	TotalCharges
0.0080 ± 0.0080	MonthlyCharges
0.0045 ± 0.0049	PaperlessBilling
0.0032 ± 0.0030	SeniorCitizen
0.0029 ± 0.0025	PaymentMethod_Electronic check
0.0027 ± 0.0013	StreamingTV
0.0022 ± 0.0026	MultipleLines
0.0019 ± 0.0013	Contract_One year
0.0015 ± 0.0039	OnlineSecurity
0.0010 ± 0.0038	InternetService_No
0.0007 ± 0.0020	OnlineBackup
0.0004 ± 0.0035	gender
0.0003 ± 0.0022	DeviceProtection
0.0002 ± 0.0023	Contract_Two year
-0.0001 ± 0.0015	PhoneService
-0.0001 ± 0.0030	StreamingMovies
-0.0009 ± 0.0026	TechSupport
	… 6 more …

Figure 34 Таблица весов признаков

Таким образом, наилучшая по эффективности модель – Random Forest Classifier с точность в ~79,5%. Хуже всего справился алгоритм Decision Tree Classifier с точностью ~ 72,6%.

Можно сделать вывод, что для повышения точности моделей, поможет большее количество данных и новые признаки. Помимо того, получилось извлечь пару инсайтов на этапе предварительного анализа. Определить тесную взаимосвязь параметра tenure и MonthlyCharge целевой переменной можно было до построения предиктивной модели.[4]

Были изучены различные методы предварительного анализа данных и построения признаков в задачах предсказания оттока клиентов. Сформировано примерное представление о том, в каких направлениях стоит продумывать решения в похожих постановках задачи.

В случае решения такой задачи в реальном кейсе, упор нужно делать на подбор гиперпараметров. Для этого процесса нужны сильные вычислительные мощности, так как этот процесс займёт большую часть времени. При этом, проделав всевозможные действия с данными, тюнинг гиперпараметров – единственный правильный вариант.[5]

7. Список использованной литературы

1. Андреас Мюллер, Введение в машинное обучение с помощью Python [Руководство для специалистов по работе с данными] / Андреас Мюллер, Сара Гвидо, – Москва, 2016-2017. – 393 с.
2. Воронцов К.В. Лекции по методу опорных векторов [Электронный ресурс]. URL: http://www.ccas.ru/voron/download/SVM.pdf (дата обращения 10.11.2020)
3. Hyperparameter Optimization With Random Search and Grid Search (machinelearningmastery.com)
4. Customer Churn Prediction in Telecommunication A Decade Review and Classification Nabgha Hashmi, Naveed Anwer Butt and Dr.Muddesar Iqbal Сентябрь 2013 (PDF) Customer Churn Prediction in Telecommunication A Decade Review and Classification (researchgate.net)
5. Predicting Potential Banking Customer Churn using Apache Spark ML and MLlib Packages: Comparative Study Hend Sayed, Manal A. Abdel-Fattah, Sherif Kholief 11.2018

Приложение (Код)

```
pip install eli5
from google.colab import drive
drive.mount("/content/drive")
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier, XGBRFClassifier
from xgboost import plot_tree, plot_importance
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, roc_curve
from sklearn import preprocessing
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV
from sklearn.feature_selection import RFE
import inspect
import seaborn as sns
import eli5
from eli5.sklearn import PermutationImportance
df = pd.read_csv('drive/MyDrive/Colab Notebooks/CustomerChurn.csv')
df.head(5)
df.drop('customerID',axis='columns',inplace=True)
df.dtypes
df.TotalCharges.values
pd.to_numeric(df.TotalCharges)
pd.to_numeric(df.TotalCharges,errors='coerce').isnull()
df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]
df.shape
df.iloc[488].TotalCharges

df[df.TotalCharges!=' '].shape
df1 = df[df.TotalCharges!=' ']
df1.shape
df1.dtypes
df1.TotalCharges = pd.to_numeric(df1.TotalCharges)
df1.TotalCharges.values
df1[df1.Churn=='No']
tenure_churn_no = df1[df1.Churn=='No'].tenure
tenure_churn_yes = df1[df1.Churn=='Yes'].tenure
plt.xlabel("tenure")
plt.ylabel("Number Of Customers")
plt.title("Customer Churn Prediction Visualiztion")
plt.hist([tenure_churn_yes, tenure_churn_no], rwidth=0.95,
color=['green','red'],label=['Churn=Yes','Churn=No'])
plt.legend()
mc_churn_no = df1[df1.Churn=='No'].MonthlyCharges
mc_churn_yes = df1[df1.Churn=='Yes'].MonthlyCharges
plt.xlabel("Monthly Charges")
```

```
plt.ylabel("Number Of Customers")
plt.title("Customer Churn Prediction Visualiztion")
plt.hist([mc_churn_yes, mc_churn_no], rwidth=0.95,
color=['green','red'],label=['Churn=Yes','Churn=No'])
plt.legend()
def print_unique_col_values(df):
       for column in df:
            if df[column].dtypes=='object':
                print(f'{column}: {df[column].unique()}')
print_unique_col_values(df1)
df1.replace('No internet service','No',inplace=True)
df1.replace('No phone service','No',inplace=True)
print_unique_col_values(df1)
yes_no_columns =
['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','
OnlineBackup',

'DeviceProtection','TechSupport','StreamingTV','StreamingMovies','Paperle
ssBilling','Churn']
for col in yes_no_columns:
    df1[col].replace({'Yes': 1,'No': 0},inplace=True)
df1['gender'].replace({'Male': 1, 'Female': 0}, inplace = True)
df1
df2 = pd.get_dummies(data=df1,
columns=['InternetService','Contract','PaymentMethod'])
df2.columns
df2.sample(5)
df2.dtypes
cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']

scaler = MinMaxScaler()
df2[cols_to_scale] = scaler.fit_transform(df2[cols_to_scale])
for col in df2:
    print(f'{col}: {df2[col].unique()}')
X = df2.drop('Churn',axis='columns')
y = df2['Churn']
models = [GaussianNB(),
          SGDClassifier(max_iter=5000, random_state=0),
```

```
          KNeighborsClassifier(n_neighbors=19),
          DecisionTreeClassifier(),
          RandomForestClassifier(n_estimators=1000,
                                 max_depth=10, random_state=0),
          SVC(decision_function_shape="ovo"),
          LogisticRegression(random_state=0,
                             solver='lbfgs', multi_class='multinomial'),
          XGBClassifier(n_estimators=1000,
                        learning_rate=0.05)]
model_names = ['Naive Bayes',
               'Stochastic Gradient Descent',
               'KNN', 'Decision trees',
               'Random Forest', 'Support Vector Machine',
               'Logistic Regression',
               'Cross Gradient Booster']
%time
def pipeline(X, y):
    mx = 0
    mxmdl = 0
    X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        test_size=0.3,
                                                        random_state=42)
    for i, model in enumerate(models):
        %time
        model.fit(X_train, y_train)
        a = accuracy_score(y_test, model.predict(X_test))
        if a > mx:
            mx = a
            mxmdl = model
            mxmdl_name = model_names[i]
        print(model_names[i], a)
    print(mxmdl_name, ' has the best accuracy of ', mx)
    return(mxmdl, mxmdl_name, mx)
fmodel = pipeline(X, y)[0]
fmodel
gs = GridSearchCV(estimator = fmodel,
```

```
                    param_grid = {'n_estimators': np.arange(900, 1100, 50),
'criterion': ['gini', 'entropy']},
                   cv = 4, verbose = 2)
br = gs.fit(X_train, y_train)

accuracy_score(y_test, br.predict(X_test))
n_estimators = np.arange(900, 1100, 25)
max_features = ['auto', 'sqrt']
max_depth = np.arange(10, 120, 10)
np.append(max_depth, None)
bootstrap = [True, False]


random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'bootstrap': bootstrap}
rf_random = RandomizedSearchCV(estimator = fmodel,
                               param_distributions = random_grid,
                               n_iter = 100,
                               cv = 3,
                               verbose=2,
                               random_state=42,
                               n_jobs = -1)
rf_random.fit(X_train, y_train)
accuracy_score(y_test, rf_random.predict(X_test))
accuracy_score(y, fmodel.predict(X))
preds = fmodel.predict(X)
print('Точность', ':', round(accuracy_score(y, preds), 5), '\n')

# Матрица сравнений
confusion_matr = confusion_matrix(y, preds, normalize='all') #normalize =
'true'
plt.figure(figsize = (16, 9))
sns.heatmap(confusion_matr, annot=True, xticklabels= ['НО', 'ВП'],
yticklabels=['ВО', 'НП'])
plt.savefig("conf matrix")
print('', confusion_matrix(y, preds, normalize='all')[0])
```

```
#Посмотрим на отдельные весы значимости признаков
perm = PermutationImportance(estimator=fmodel, random_state=1)
perm.fit(X_test, y_test)
eli5.show_weights(estimator=perm, feature_names = X_test.columns.tolist())
```