ПРОГНОЗИРОВАНИЕ ОТТОКА КЛИЕНТОВ ТЕЛЕКОМ-ОПЕРАТОР

КСАО-01-19

Груненков Д.В.

ПОСТАНОВКА ЗАДАЧИ

- С проблемой оттока клиентов сталкиваются все компании, предоставляющие товары или услуги. В связи с 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.
- 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).

- Для обучения моделей будем использовать набор данных telecom_churn, загруженный с Kaggle. Каждая строка базы представляет собой информацию по одному клиенту – это объект исследования. Столбцы – признаки объекта.

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
128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	Fals
107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	Fals
137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	Fals
84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	Fals
75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	Fals

Считаем данные из файла и выведем первые пять строк.

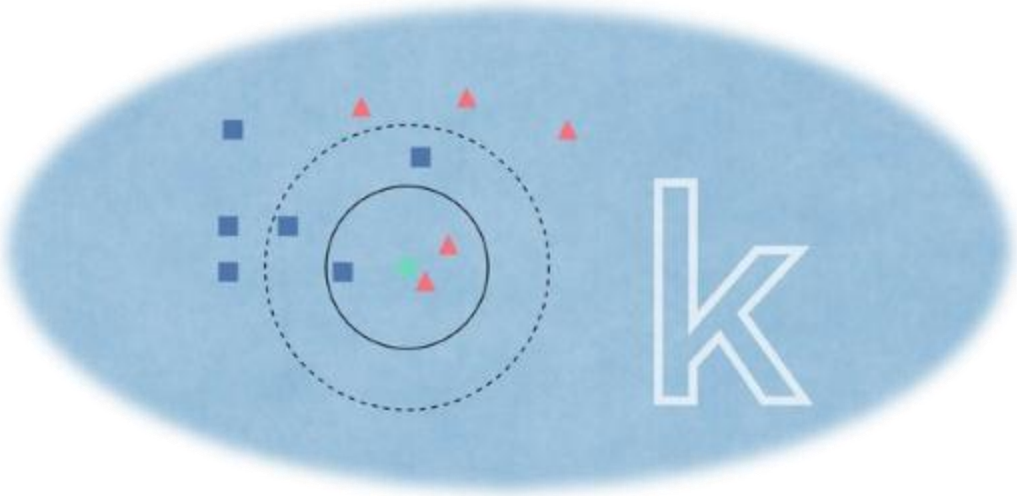
```
# чтение данных из файла
df = pd.read_csv('telecom_churn.csv')
# выведем первые 5 строк
df.head(5).
# выведем размер датасета
print(df.shape)
```

В таблице 3333 строки и 20 столбцов.

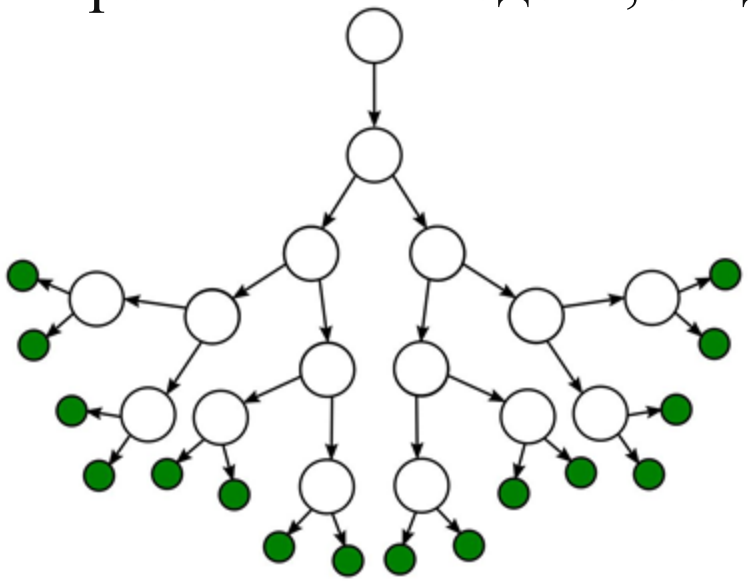
- Применим метод astype к признаку Churn и переведём его в int64:
- *Df[**‘Churn’**] = df[**‘Churn’**].astype(**‘int64’**)*
- Для категориальных (тип object) и булевых (тип bool) признаков можно воспользоваться методом value_counts. Посмотрим на распределение данных по нашей целевой переменной — Churn:
- *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 - 0.1449144914491492.*
- Это означает, что отток клиентов компании составляет 14.5%.

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.
- Суть метода k-ближайших соседей проста: посмотри на соседей вокруг, какие из них преобладают, таковым ты и являешься. Формально основой метода является гипотеза компактности: если метрика расстояния между примерами введена удачно, то схожие примеры гораздо чаще лежат в одном классе, чем в разных.



- Дерево решений представляет собой иерархическую древовидную структуру, состоящую из правила вида «Если …, то ...». За счет обучающего множества правила генерируются автоматически в процессе обучения.
- В отличие от нейронных сетей, деревья как аналитические модели проще, потому что правила генерируются на естественном языке: например, «Если реклама привела 1000 клиентов, то она настроена хорошо».
- В обучающем множестве для примеров должно быть задано целевое значение, так как деревья решений — модели, создаваемые на основе обучения с учителем.



- Представленные 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.
- 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)
```

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 – 11.

```
In [161]: tree_grid.best_params_
Out[161]: {'max_depth': 6, 'max_features': 11}

In [162]: tree_grid.best_score_
Out[162]: 0.9442841256858221

In [163]: accuracy_score(y_holdout, tree_grid.predict(X_holdout))
Out[163]: 0.948
```

- Теперь настроим число соседей в алгоритме kNN. Наилучшие результаты дает модель с 7 соседями

```
Out[158]: ({'knn__n_neighbors': 7}, 0.8859867109023905)

B [159]: accuracy_score(y_holdout, knn_grid.predict(X_holdout))

Out[159]: 0.89
```

ВЫВОДЫ

- 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.