GeekBrains, ML in Business

Lesson 1 Homework

Автор материала: Зраев Артем.

Можно использовать в каких угодно целях.

В задании нужно загрузить датасет с данными оттока и ответить на несколько вопросов (написать код). При этом сам датасет уже есть и его необязательно качать с репозитория

Цель задания: проверить базовые навыки работы студентов с Pandas, умение проводить такой же базовый EDA (exploratory data analysis), делать feature engineering и обучать и валидировать модель.

Список столбцов с типами данных в датасете:

- customerID object
- 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

In [1]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("./WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.head(3)
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	Te
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	

3 rows × 21 columns

1. Какое соотношение мужчин и женщин в представленном наборе данных?

In [2]:
```
# В процентах (%)
df['gender'].value_counts() * 100 / df['gender'].shape[0]
```

Out[2]:
```
Male      50.47565
Female    49.52435
Name: gender, dtype: float64
```

2. Какое количество уникальных значений у поля InternetService?

In [3]:
```
df['InternetService'].unique().shape[0]
```

Out[3]: 3

3.

3. Выведите статистики по полю TotalCharges (median, mean, std).

In [4]:
```
df['TotalCharges'].describe()
```

Out[4]:
```
count     7043
unique    6531
top       20.2
freq        11
Name: TotalCharges, dtype: object
```

В чем странность того, что вы получили? (подсказка: смотреть нужно на тип данных)

Метод не срабатывает, как можно было ожидать, т.к. тип столбца - object.

4. Сделайте замену значений поля PhoneService на числовые (Yes->1, No->0)

In [5]:
```
def yes_to_one(string: str) -> int:
    return 1 if string == 'Yes' else 0
```

In [6]:
```
df['PhoneService'] = df['PhoneService'].map(yes_to_one)
```

In [7]:
```
df['PhoneService']
```

Out[7]:
```
0       0
1       1
2       1
3       0
4       1
       ..
7038    1
7039    1
7040    0
7041    1
7042    1
Name: PhoneService, Length: 7043, dtype: int64
```

5. Сделайте замену пробелов в поле TotalCharges на np.nan и приведите поле к типу данных float32. Затем заполните оставшиеся пропуски значением 0 с помощью метода fillna у столбца. Снова выведите статистики и сравните с тем, что вы видели в вопросе 3

In [8]:
```
df.loc[df['TotalCharges'] == ' ', 'TotalCharges'] = np.nan
df.loc[df['TotalCharges'] == ' ']
```

Out[8]:

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	Tec

0 rows × 21 columns

In [9]:
```
df['TotalCharges'] = df['TotalCharges'].astype(np.float32)
```

In [10]:
```
df['TotalCharges'] = df['TotalCharges'].fillna(0)
df['TotalCharges'].isna().sum()
```

Out[10]:
```
0
```

In [11]:
```
df['TotalCharges'].describe()
```

Out[11]:
```
count    7043.000000
mean     2279.734375
std      2266.794434
min         0.000000
25%       398.549988
50%      1394.550049
75%      3786.599976
max      8684.799805
Name: TotalCharges, dtype: float64
```

Медиана = 1394.55, среднее = 2279.73, std = 2266.79.

6. Сделайте замену значений поля Churn на числовые (Yes -> 1, No - 0)

In [12]:
```
df['Churn'] = df['Churn'].map(yes_to_one)
```

In [13]:
```
df['Churn']
```

Out[13]:
```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
```

```
7042    0
Name: Churn, Length: 7043, dtype: int64
```

7. Сделайте замену значений полей StreamingMovies, StreamingTV, TechSupport на числовые (Yes -> 1, No -> 0, No internet service->0)

In [14]:
```
df[['StreamingMovies', 'StreamingTV', 'TechSupport']] = df[['StreamingMovies', 'StreamingTV', 'TechSupport']].applymap(yes_to_one)
```

In [15]:
```
df[['StreamingMovies', 'StreamingTV', 'TechSupport']]
```

Out[15]:

	StreamingMovies	StreamingTV	TechSupport
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	1
4	0	0	0
...	...	...	...
7038	1	1	1
7039	1	1	0
7040	0	0	0
7041	0	0	0
7042	1	1	1

7043 rows × 3 columns

8. Заполните пропуски в поле PhoneService значением 0

In [16]:
```
df['PhoneService'] = df['PhoneService'].fillna(0)
df['PhoneService'].isna().sum()
```

Out[16]: 0

8. Для нашего датасета оставьте только указанный ниже список полей, удалив все другие и выведите верхние 3 строки

In [17]:
```
columns = ['gender', 'tenure', 'PhoneService', 'TotalCharges',
           'StreamingMovies', 'StreamingTV', 'TechSupport', 'Churn']

df = df[columns]
df.head(3)
```

Out[17]:

	gender	tenure	PhoneService	TotalCharges	StreamingMovies	StreamingTV	TechSupport	Churn
0	Female	1	0	29.850000	0	0	0	0
1	Male	34	1	1889.500000	0	0	0	0
2	Male	2	1	108.150002	0	0	0	1

9. Разделите датасет на тренировочную и тестовую выборку (подсказка - воспользуйтесь train_test_split из sklearn.model_selection. Ссылка - https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html)

In [18]:
```
from sklearn.model_selection import train_test_split

features = ['gender', 'tenure', 'PhoneService', 'TotalCharges', 'StreamingMovies', 'StreamingTV', 'TechSupport']
target = 'Churn'

X_train, X_test, y_train, y_test = train_test_split(df[features], df[target], test_size=0.3)
```

In [19]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[19]: ((4930, 7), (4930,), (2113, 7), (2113,))

10. соберите pipeline для поля gender (нужно разобраться и изучить https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html) из классов ColumnSelector и OHEEncoder, которые уже написаны ниже заранее

In [20]:
```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.pipeline import Pipeline

class ColumnSelector(BaseEstimator, TransformerMixin):
    """
    Transformer to select a single column from the data frame to perform additional transformations on
    """
    def __init__(self, key):
        self.key = key

    def fit(self, X, y=None):
```

```
        return self

    def transform(self, X):
        return X[self.key]

class NumberSelector(BaseEstimator, TransformerMixin):
    """
    Transformer to select a single column from the data frame to perform additional transformations on
    Use on numeric columns in the data
    """
    def __init__(self, key):
        self.key = key

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X[[self.key]]

class OHEEncoder(BaseEstimator, TransformerMixin):
    def __init__(self, key):
        self.key = key
        self.columns = []

    def fit(self, X, y=None):
        self.columns = [col for col in pd.get_dummies(X, prefix=self.key).columns]
        return self

    def transform(self, X):
        X = pd.get_dummies(X, prefix=self.key)
        test_columns = [col for col in X.columns]
        for col_ in test_columns:
            if col_ not in self.columns:
                X[col_] = 0
        return X[self.columns]

gender = Pipeline([
                ('selector', ColumnSelector(key='gender')),
                ('ohe', OHEEncoder(key='gender'))
            ])
```

11. Вызовите метод fit_transform у пайплайна gender и передайте туда нашу тренировочную выборку (пример по ссылке из документации https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline.fit)

In [21]:
```
gender.fit_transform(X_train, y_train)
```

Out[21]:

	gender_Female	gender_Male
2969	1	0
4904	1	0
2447	0	1
91	0	1
825	1	0
...	...	...
5510	0	1
1196	0	1
3713	0	1
4738	0	1
836	0	1

4930 rows × 2 columns

12. Здесь код писать уже не нужно (все сделано за вас). К полю tenure применяем StandardScaler (нормируем и центрируем). Ссылка - https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

Вопрос - в каких случаях это может быть полезно?

In [22]:
```
from sklearn.preprocessing import StandardScaler

tenure =  Pipeline([
                ('selector', NumberSelector(key='tenure')),
                ('standard', StandardScaler())
            ])
```

Стандартизация улучшает точность или даже требуется для многих алгоритмов машинного обучения.

Пример - KNN, SVM, регрессия работают значительно лучше при стандартизированных данных (более похожих на Гауссово нормальное распределение).

13. Напишите аналогичный (как для tenure) преобразователь поля TotalCharges

In [23]:
```
total_charges = Pipeline([
```

```
    ('selector', NumberSelector(key='TotalCharges')),
    ('standardization', StandardScaler())
])
```

In [ ]:

Объединение всех "кубиков" очень легко сделать таким образом

In [24]:
```
from sklearn.pipeline import FeatureUnion

number_features = Pipeline([
                ('selector', ColumnSelector(key=['PhoneService',
                                                 'StreamingMovies', 'StreamingTV',
                                                 'TechSupport']))
            ])
```

In [25]:
```
feats = FeatureUnion([('tenure', tenure),
                      ('TotalCharges', total_charges),
                      ('continuos_features', number_features),
                      ('gender', gender)])
feature_processing = Pipeline([('feats', feats)])
```

На этом этапе что мы сделали:

1. написали преобразователь поля gender, который делает OHE кодирование
2. написали преобразователь для поля tenure, который нормирует и центрирует его
3. повторили п. 2 для поля TotalCharges
4. для всех остальных просто взяли признаки как они есть, без изменений

У нас уже готов наш пайплайн, который преобразовывает признаки. Давайте обучим модель поверх него. В качестве модели возьмем RandomForestClassifier

In [26]:
```
from sklearn.ensemble import RandomForestClassifier

pipeline = Pipeline([
    ('features',feats),
    ('classifier', RandomForestClassifier(random_state = 42)),
])

pipeline.fit(X_train, y_train)
```

Out[26]:
```
Pipeline(steps=[('features',
                 FeatureUnion(transformer_list=[('tenure',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='tenure')),
                                                                 ('standard',
                                                                  StandardScaler())])),
                                                ('TotalCharges',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='TotalCharges')),
                                                                 ('standardization',
                                                                  StandardScaler())])),
                                                ('continuos_features',
                                                 Pipeline(steps=[('selector',
                                                                  ColumnSelector(key=['PhoneService',
                                                                                      'StreamingMovies',
                                                                                      'StreamingTV',
                                                                                      'TechSupport']))])),
                                                ('gender',
                                                 Pipeline(steps=[('selector',
                                                                  ColumnSelector(key='gender')),
                                                                 ('ohe',
                                                                  OHEEncoder(key='gender'))]))])),
                ('classifier', RandomForestClassifier(random_state=42))])
```

14. Сделайте прогноз вероятности оттока для X_test с помощью нашего предобученного на предыдущем шаге пайплайна и убедитесь что вам возвращаются вероятности для 2 классов

In [27]:
```
prediction = pipeline.predict_proba(X_test)
prediction
```

Out[27]:
```
array([[0.74      , 0.26      ],
       [0.99      , 0.01      ],
       [0.98      , 0.02      ],
       ...,
       [0.75      , 0.25      ],
       [0.59715618, 0.40284382],
       [1.        , 0.        ]])
```

15. Посчитайте метрики качества получившейся модели (roc_auc, logloss)

In [28]:
```
from sklearn.metrics import roc_auc_score, log_loss
```

In [29]:
```
roc_auc_score(y_test, prediction[:, 1])
```

Out[29]:
```
0.7767111088917741
```

In [30]:
```
log_loss(y_test, prediction[:, 1])
```

Out[30]: 0.7311507214827245

Сохраним наш пайплайн

In [31]:
```
import dill
with open("model_RF.dill", "wb") as f:
    dill.dump(pipeline, f)
```