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

Ввод [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	TechSupport	Streami
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	

3 rows × 21 columns

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

Ввод [2]:
```
#Ваш код здесь
print(f"Доля мужчин: {sum(df['gender'] == 'Male')/df.shape[0]}")
print(f"Доля женщин: {sum(df['gender'] == 'Female')/df.shape[0]}")
```

```
Доля мужчин: 0.504756495811444
Доля женщин: 0.495243504188556
```

2. Какое количество уникальных значений у поля InternetService?

Ввод [3]:
```
#Ваш код здесь
print(f"Количество уникальных значений поля InternetService: {df['InternetService'].unique().shape[0]}")
```

```
Количество уникальных значений поля InternetService: 3
```

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

Ввод [4]:
```
#Ваш код здесь
print('median: ', df['TotalCharges'].median())
print('mean: ', df['TotalCharges'].mean())
print('std: ', df['TotalCharges'].std())
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
T:\Anaconda\lib\site-packages\pandas\core\nanops.py in nanmedian(values, axis, skipna, mask)
    686         try:
--> 687             values = values.astype("f8")
    688         except ValueError as err:

ValueError: could not convert string to float: ''

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
<ipython-input-4-aaeceec34538> in <module>
      1 #Ваш код здесь
----> 2 print('median: ', df['TotalCharges'].median())
      3 print('mean: ', df['TotalCharges'].mean())
      4 print('std: ', df['TotalCharges'].std())

T:\Anaconda\lib\site-packages\pandas\core\generic.py in median(self, axis, skipna, level, numeric_only, **kwargs)
  11173             self, axis=None, skipna=None, level=None, numeric_only=None, **kwargs
  11174         ):
> 11175             return NDFrame.median(self, axis, skipna, level, numeric_only, **kwargs)
  11176
  11177         # pandas\core\generic.py:10975: error: Cannot assign to a method

T:\Anaconda\lib\site-packages\pandas\core\generic.py in median(self, axis, skipna, level, numeric_only, **kwargs)
  10729
  10730     def median(self, axis=None, skipna=None, level=None, numeric_only=None, **kwargs):
> 10731         return self._stat_function(
  10732             "median", nanops.nanmedian, axis, skipna, level, numeric_only, **kwargs
  10733         )

T:\Anaconda\lib\site-packages\pandas\core\generic.py in _stat_function(self, name, func, axis, skipna, level, numeric_only, *
*kwargs)
  10709         if level is not None:
  10710             return self._agg_by_level(name, axis=axis, level=level, skipna=skipna)
> 10711         return self._reduce(
  10712             func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_only
  10713         )

T:\Anaconda\lib\site-packages\pandas\core\series.py in _reduce(self, op, name, axis, skipna, numeric_only, filter_type, **kwd
s)
   4180                 )
   4181             with np.errstate(all="ignore"):
-> 4182                 return op(delegate, skipna=skipna, **kwds)
   4183
   4184     def _reindex_indexer(self, new_index, indexer, copy):

T:\Anaconda\lib\site-packages\pandas\core\nanops.py in f(values, axis, skipna, **kwds)
```

```
    133                     result = alt(values, axis=axis, skipna=skipna, **kwds)
    134             else:
--> 135                 result = alt(values, axis=axis, skipna=skipna, **kwds)
    136
    137             return result

T:\Anaconda\lib\site-packages\pandas\core\nanops.py in nanmedian(values, axis, skipna, mask)
    688         except ValueError as err:
    689             # e.g. "could not convert string to float: 'a'"
--> 690             raise TypeError from err
    691         if mask is not None:
    692             values[mask] = np.nan
```

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

Тип данных - object. Числа хронятся в строковом виде.

- При определении медианы появляется ошибка перевода пустой строки в число (float).
- При определении среднего происходит конкотенация строк с дальнейшим переводом получившейся строки в число с плавающей запятой, что не возможно, и поэтому появляется ошибка.
- При подсчете стандартного отклонения появляется ошибка перевода пустой строки в число (float).

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

Ввод [5]:
```
#Ваш код здесь
df.replace({'PhoneService': {'Yes': 1, 'No': 0}}, inplace=True)
df['PhoneService']
```

Out[5]:
```
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

Ввод [6]:
```
#Ваш код здесь
df.replace({'TotalCharges': {' ': np.nan}}, inplace=True)
df['TotalCharges'] = df['TotalCharges'].astype(np.float32)
df['TotalCharges'].fillna(0, inplace=True)

print('median: ', df['TotalCharges'].median())
print('mean: ', df['TotalCharges'].mean())
print('std: ', df['TotalCharges'].std())
# Появились результаты
```

```
median:  1394.550048828125
mean:  2279.732177734375
std:  2266.79443359375
```

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

Ввод [7]:
```
#Ваш код здесь
df.replace({'Churn': {'Yes': 1, 'No': 0}}, inplace=True)
df['Churn']
```

Out[7]:
```
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)

Ввод [8]:
```
#Ваш код здесь
replace_dict = {'Yes': 1, 'No': 0, 'No internet service': 0}
df.replace({'StreamingMovies': replace_dict, 'StreamingTV': replace_dict, 'TechSupport': replace_dict}, inplace=True)
df['StreamingTV']
```

Out[8]:

```
0       0
1       0
2       0
3       0
4       0
      ..
```

8. Заполните пропуски в поле PhoneService значением 0

Ввод [9]:
```
#Ваш код здесь
df['PhoneService'].fillna(0, inplace=True)
df['PhoneService']
```

Out[9]:
```
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
```

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

Ввод [10]:
```
columns = ['gender', 'tenure', 'PhoneService', 'TotalCharges',
           'StreamingMovies', 'StreamingTV', 'TechSupport', 'Churn']
#Ваш код здесь
df = df[columns]
df.head(3)
```

Out[10]:

	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 (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html))

Ввод [11]:
```
from sklearn.model_selection import train_test_split

features = ['gender', 'tenure', 'PhoneService', 'TotalCharges', 'StreamingMovies', 'StreamingTV', 'TechSupport']
target = 'Churn'
#Ваш код здесь
X = df[features]
y = df[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0, stratify=y)
```

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

Ввод [12]:
```
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 (https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline.fit))

Ввод [13]:
```
#Ваш код здесь
gender.fit_transform(X_train)
```

Out[13]:

	gender_Female	gender_Male
5229	1	0
5668	1	0
1667	0	1
6709	0	1
3609	1	0
...	...	...
1712	1	0
1218	0	1
525	0	1
5748	1	0
6513	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 (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html)

Вопрос - в каких случаях это может быть полезно?

Ввод [14]:
```
from sklearn.preprocessing import StandardScaler

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

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

Ввод [15]:
```
#Ваш код здесь
total_charges = Pipeline([
                    ('selector', NumberSelector(key='TotalCharges')),
                    ('standard', StandardScaler())
                ])
```

Объединение всех "кубиков" очень легко сделать таким образом

Ввод [16]:
```
from sklearn.pipeline import FeatureUnion

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

Ввод [17]:
```
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

Ввод [18]:
```
from sklearn.ensemble import RandomForestClassifier

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

pipeline.fit(X_train, y_train)
```

Out[18]:
```
Pipeline(steps=[('features',
                 FeatureUnion(transformer_list=[('tenure',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='tenure')),
                                                                 ('standard',
                                                                  StandardScaler())])),
                                                ('TotalCharges',
                                                 Pipeline(steps=[('selector',
                                                                  NumberSelector(key='TotalCharges')),
                                                                 ('standard',
                                                                  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

Ввод [19]:
```
#Ваш код здесь
y_pred_proba = pipeline.predict_proba(X_test)
y_pred_proba
```

Out[19]:

```
array([[0.23, 0.77],
       [0.97, 0.03],
       [0.99, 0.01],
```

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

Ввод [20]:
```
from sklearn.metrics import roc_auc_score, log_loss

#Ваш код здесь
roc_auc_score(y_test, y_pred_proba[:, 1]), log_loss(y_test, y_pred_proba[:, 1])
```

Out[20]:
```
(0.7592514747229726, 1.003998593432096)
```

Сохраним наш пайплайн

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

Ввод [ ]:
```
```