Перед вами [ноутбук](https://drive.google.com/file/d/1dLiD4SXJI_ue8X-gKLIFNtWJCdPTor-z/view?usp=sharing) с примером подготовки данных и тренировки нескольких моделей машинного обучения. В нём использован уже знакомый вам [датасет с информацией про автомобили](https://drive.google.com/file/d/14Zzg2DrIfv5DG5UQCM7zVP-SHtz2YxOV/view?usp=sharing), вы работали с ним в предыдущих модулях.

*Задача, как и раньше, — многоклассовая классификация. Вам предстоит предсказать категорию цены (high, medium или low), за которую можно продать автомобиль.*

В ноутбуке нет ничего нового, вы уже это делали. Перед тем как приступать к заданию, разберитесь с кодом и убедитесь, что всё понятно.

Цели практической работы

Реализовать пайплайн машинного обучения и закрепить новые умения из модуля.

Что нужно сделать

Повторите все шаги, которые сделаны [в ноутбуке](https://drive.google.com/file/d/1dLiD4SXJI_ue8X-gKLIFNtWJCdPTor-z/view?usp=sharing), но оформите их в виде sklearn-пайплайна:

1. Создайте новый проект в PyCharm.
2. Напишите необходимый код в файле main.py.
3. Когда закончите и соберёте рабочий пайплайн обработки данных, обучите три модели (те же, что тренируются в ноутбуке):
   1. оформите обучение классификатора в виде итогового шага пайплайна;
   2. выберите лучшую модель на кросс-валидации (по метрике accuracy) и сохраните итоговый пайплайн в pickle-файл.

Пайплайн тут сложнее того, что разобран в видео. Шаги пайплайна:

1. Удаление ненужных колонок (фильтрация датасета).
2. Сглаживание (удаление) выбросов в колонке year.
3. Создание новых предикторов (short\_model и age\_category).
4. Заполнение пропусков в численных признаках медианой и масштабирование в StandardScaler.
5. Заполнение пропусков в категориальных переменных и кодирование OneHotEncoder.

Шаги 1–3

Разберитесь самостоятельно, как реализовать шаги с первого по третий.

***Подсказка 1*:** [класс FunctionalTransformer](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.FunctionTransformer.html) (из пакета preprocessing библиотеки sklearn) умеет превращать любую функцию в преобразователь, который можно использовать как шаг в пайплайне sklearn.

Для примера рассмотрим вариант решения для первого шага нашего пайплайна:

1. Написать функцию *filter\_data*:

def filter\_data(df):  
   columns\_to\_drop = [  
       'id',  
       'url',  
       'region',  
       'region\_url',  
       'price',  
       'manufacturer',  
       'image\_url',  
       'description',  
       'posting\_date',  
       'lat',  
       'long'  
   ]  
  
   *# Возвращаем копию датафрейма, inplace тут делать нельзя!*  
*return df.drop(columns\_to\_drop, axis=1)*

Такая функция принимает на вход датафрейм, преобразует его (не inplace) и возвращает вызывающей стороне. В дальнейшем преобразованный датафрейм может попасть в следующий преобразователь вашего пайплайна и так далее.

1. Сделать функцию шагом пайплайна с помощью класса FunctionalTransformer:

preprocessor = Pipeline(steps=[  
   ('filter', FunctionTransformer(filter\_data)),  
   …  
])

Разберитесь в примере и сделайте преобразователи для шагов 2 и 3 по аналогии.

***Подсказка*2**: учтите, что функции, которые используются для преобразования данных, не должны преобразовывать датафрейм inplace. Всегда работайте с копией датафрейма (например, создавайте копию сразу на входе в функцию), иначе пайплайн будет одноразовым: если вы два раза примените его к одному и тому же датафрейму, то вряд ли получите ожидаемый результат.

Шаги 4–5

Эти шаги разобраны в видео. Все необходимые преобразователи есть в библиотеке sklearn. Составьте пайплайны для численных и категориальных фич и объедините их с помощью ColumnTransformer.

***Подсказка 3***: чтобы обращаться к категориальным и численным признакам, не имея доступа к датафрейму и не перечисляя имена колонок, используйте функцию [make\_column\_selector](https://scikit-learn.org/stable/modules/generated/sklearn.compose.make_column_selector.html) из пакета compose библиотеки sklearn. Например:

* make\_column\_selector(dtype\_include=['int64', 'float64']) — создаёт селектор числовых признаков из практически любого датасета;
* make\_column\_selector(dtype\_include=object) — создаёт селектор категориальных признаков из практически любого датасета.

Что оценивается

* Файл main.py содержит код, реализующий пайплайн со всеми шагами задания.
* Файл main.py запускается, выполняется и генерирует pickle-файл.