ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Практика решающего дерева

Аннотация к занятию: на данном занятии обучающиеся построят полный конвейер машинного обучения: от загрузки и исследования данных до метрик качества с использованием алгоритма решающего дерева.

Цель занятия: сформировать у обучающихся представление о построении полного конвейера обработки данных с использованием алгоритма решающего дерева.

Задачи занятия:

- рассмотреть три этапа работы с данными: исследование данных, подготовку данных, обучение и перебор параметров алгоритма;
- познакомить обучающихся с функциями библиотеки Sklearn;
- применить полученные знания на практике.

Ход занятия

Этап занятия	Время	Деятельность педагога	Комментарии, рекомендации для педагогов
Организационный этап	2 мин.	Добрый день! Добро пожаловать на урок	Приветствие. Создание в классе атмосферы психологического комфорта
Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся	10 мин.	На этом занятии мы построим полный конвейер обработки данных по продажам автомобилей с использованием алгоритма решающего дерева. В рамках конвейера мы рассмотрим три этапа работы с данными	Способствовать обсуждению мотивационных вопросов
Изучение нового материала	50 мин.	Мы будем работать с данными о продаже автомобилей в США. Целевая переменная — цена автомобиля. Подобные данные могут быть полезны как для автоматизированного выбора выгодных предложений, так и для построения рекомендательных систем на сайтах объявлений. Источник данных: https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes?select=bmw.csv	Для справки: Файл для работы: https://drive.google.com/file/d/1Y1bNZwulQeYE9hCWkSoiHy_R5gi4eelV/view?usp=sharing

Для удобства обработки в первой ячейке данные о продаже различных автомобилей по производителям были объединены в единый Pandas dataframe и загружены на удалённый облачный сервер. Библиотека Pandas позволяет загружать данные напрямую из облака, не скачивая их в файл:

```
1  import pandas as pd
2
3  df = pd.read_csv('https://dl.uploadgram.me/6081b6b3d5d36h?raw').drop(columns=['Unnamed: 0'])
4  df
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: Columns (3,
```

Первое после загрузки, в чём нужно убедиться, — корректность загрузки. Данные должны располагаться в отдельных колонках, названия колонок должны быть читаемые, не должно быть ошибок кодирования. В нашем случае подобных ошибок нет, датафрейм был загружен корректно.

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize	fuel type	engine size	mileage2	fuel type2	engine size2	r
0	T-Roc	2019.0	25000	Automatic	13904	Diesel	145.0	49.6	2.0	NaN	NaN	NaN	NaN	NaN	
1	T-Roc	2019.0	26883	Automatic	4562	Diesel	145.0	49.6	2.0	NaN	NaN	NaN	NaN	NaN	
2	T-Roc	2019.0	20000	Manual	7414	Diesel	145.0	50.4	2.0	NaN	NaN	NaN	NaN	NaN	
3	T-Roc	2019.0	33492	Automatic	4825	Petrol	145.0	32.5	2.0	NaN	NaN	NaN	NaN	NaN	
4	T-Roc	2019.0	22900	Semi-Auto	6500	Petrol	150.0	39.8	1.5	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
118145	I30	2016.0	8680	Manual	25906	Diesel	NaN	78.4	1.6	NaN	NaN	NaN	NaN	NaN	
118146	I40	2015.0	7830	Manual	59508	Diesel	NaN	65.7	1.7	NaN	NaN	NaN	NaN	NaN	
118147	I10	2017.0	6830	Manual	13810	Petrol	NaN	60.1	1.0	NaN	NaN	NaN	NaN	NaN	

Исследование данных
Изучая набор данных, можно посмотреть, как заполнены значения в ячейках. С помощью функции value_counts() можно узнать количество уникальных записей. Можно заметить, что названия записаны с пробелом. Подобное может быть связано как с форматом разделителя, так и с тем, что часть данных была сохранена с пробелами. Если в одном случае пробел присутствует перед Focus, а в другом нет, алгоритм будет воспринимать их как разные аргументы. Поскольку видно, что в верхних и нижних записях пробел есть, скорее всего, это связано с разделителями:

```
1  df.model.value_counts()
```

```
Focus             15590
C Class           11553
Fiesta             6557
Golf               4863
Corsa              3441
                   ...
Ranger                1
Accent                1
Escort                1
Transit Tourneo       1
Amica                 1
Name: model, Length: 195, dtype: int64
```

Используя тепловую карту, отобразим пропуски в данных:

```
1  import seaborn as sns
2  from matplotlib.pyplot import figure
3
4  figure(figsize = (10, 15))
5  sns.heatmap(df.isnull(), cbar=False)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ea1a93790>
```

Видно, что для ряда участков (белых) значения присутствуют в 10-й колонке и далее. Такое заполнение связано с тем, что в выборку попали данные по продажам автомобилей из Великобритании, что видно по колонке налога в фунтах. Поскольку большая часть выборки была собрана на основе данных о рынке США, мы отбросим все

колонки после 10 и уберём строки, содержащие как минимум один пропуск:

```
1  df.drop(columns = df.columns[9:], inplace=True)
2  df.reset_index(inplace=True)
3  df.dropna(inplace=True)
4
5  figure(figsize = (10, 15))
6  sns.heatmap(df.isnull(), cbar=False)
7  df
```

При отображении тепловой карты и с помощью функции info видно, что теперь пропусков нет:

```
[ ]  1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 94327 entries, 0 to 113289
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   index         94327 non-null  int64
 1   model         94327 non-null  object
 2   year          94327 non-null  float64
 3   price         94327 non-null  object
 4   transmission  94327 non-null  object
 5   mileage       94327 non-null  object
 6   fuelType      94327 non-null  object
 7   tax           94327 non-null  float64
 8   mpg           94327 non-null  float64
 9   engineSize    94327 non-null  float64
dtypes: float64(4), int64(1), object(5)
memory usage: 7.9+ MB
```

Рассмотрим описательные статистики числовых колонок:

```
1  df.describe()
```

	index	year	tax	mpg	engineSize
count	94327.000000	94327.000000	94327.000000	94327.000000	94327.000000
mean	56719.836346	2017.086698	120.256183	55.235816	1.673861
std	35046.756072	2.133897	63.404805	16.291667	0.562523
min	0.000000	1970.000000	0.000000	0.300000	0.000000
25%	23581.500000	2016.000000	125.000000	47.100000	1.300000
50%	60672.000000	2017.000000	145.000000	54.300000	1.600000
75%	89707.500000	2019.000000	145.000000	62.800000	2.000000
max	113289.000000	2060.000000	580.000000	470.800000	6.600000

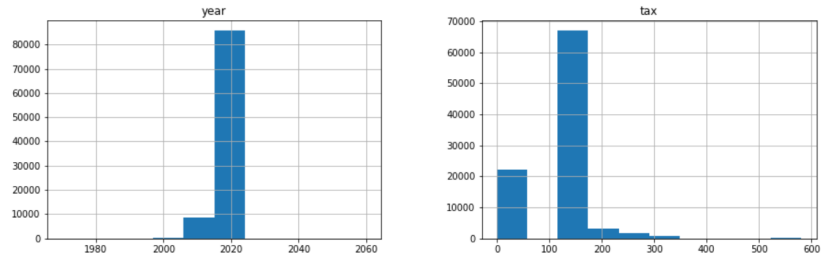
Видно, что в колонке даты выпуска автомобиля максимальное значение — 2060 год, что явно невозможно. В дальнейшем удалим такие выбросы.

Отбросим колонку «Индекс», полученную в результате отброса британских автомобилей. Так у нас не будет лишних значений номера, не нужных в процессе обучения. Визуализируем значения на гистограмме, чтобы убедиться, что у нас нет аномалий. Среди аномальных значений (например, у признака расхода топлива — mpg) незначительное число больших.

```
[ ]  1  del df['index']
```

```
[ ]  1  df.hist(figsize=(15, 10))
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6e9d8f1590>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6e9d890e90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6e9d8534d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6e9d80aad0>]],
      dtype=object)
```



Применим к названию модели функцию strip. Она убирает лишние пробелы:

```
[ ]  1  df['model'] = df['model'].apply(lambda x: x.strip())
```

Уберём единственную запись с автомобилем из будущего (некорректно заполненную):

```
[ ]  1  df.year.describe()
```

```
count    94327.000000
mean      2017.086698
std          2.133897
min       1970.000000
25%       2016.000000
50%       2017.000000
75%       2019.000000
max       2060.000000
Name: year, dtype: float64
```

```
[ ]  1  df[df['year'] >= 2021]
```

	model	year	price	transmission	mileage	fuelType	tax	mpg	engineSize
99419	Fiesta	2060.0	6495	Automatic	54807	Petrol	205.0	42.8	1.4

```
[ ]  1  df = df[df['year'] < 2021]
     2  df
```

Снова посмотрим на описательные статистики. Видно, что в числовых колонках отсутствует price — цена. Помимо этого, можно сэкономить немного памяти при обработке признаков mpg и mileage, поскольку они и так всегда целые. Год можно представить в виде категориального признака т.к. в таком случае модели будет проще определить ценовой диапазон авто.

```
[ ]  1  df.describe()
```

	year	tax	mpg	engineSize
count	94326.000000	94326.000000	94326.000000	94326.000000
mean	2017.086243	120.255285	55.235948	1.673864
std	2.129329	63.404540	16.291703	0.562525
min	1970.000000	0.000000	0.300000	0.000000
25%	2016.000000	125.000000	47.100000	1.300000
50%	2017.000000	145.000000	54.300000	1.600000
75%	2019.000000	145.000000	62.800000	2.000000
max	2020.000000	580.000000	470.800000	6.600000

Перекодировка позволит модели упростить обработку

```
[ ]  1  df['price'] = df['price'].astype('int')
     2  df['year'] = df['year'].astype('object')
     3  df['mpg'] = df['mpg'].astype('int')
     4  df['mileage'] = df['mileage'].astype('int')
```

Кодирование

Рассмотрим два варианта кодирования. Первый — label encoder. Он предполагает, что для каждой категориальной

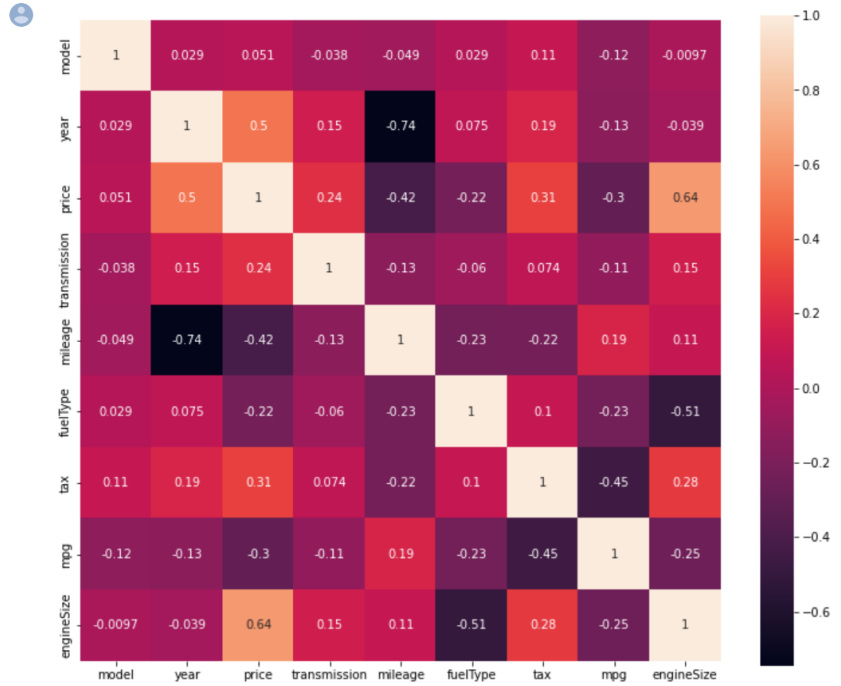
колонки в соответствие будет поставлена колонка с числами для каждого уникального значения (от 0 до n, где n — количество уникальных записей). Такую кодировку будет производить функция LabelEncoder библиотеки Sklearn. Каждый Label Encoder каждой категориальной колонки будет записан функцией в словарь. Обращаясь к ней, перекодируем ими колонки и запишем в копию датафрейма:

```python
# Подключаем класс для предобработки данных
from sklearn import preprocessing
import numpy as np

# Напишем функцию, которая принимает на вход DataFrame, кодирует числовыми значениями категориальные признаки
# и возвращает обновленный DataFrame и сами кодировщики.
def number_encode_features(init_df):
    result = init_df.copy() # копируем нашу исходную таблицу
    encoders = {}
    for column in result.columns:
        if result.dtypes[column] == np.object: # np.object -- строковый тип / если тип столбца - строка, то нужно его закодировать
            encoders[column] = preprocessing.LabelEncoder() # для колонки column создаем кодировщик
            result[column] = encoders[column].fit_transform(result[column]) # применяем кодировщик к столбцу и перезаписываем столбец
    return result, encoders

# используем все, кроме описания
encoded_data, encoders = number_encode_features(df) # Теперь encoded data содержит закодированные кат. признаки
encoded_data
```

Отобразим тепловую карту корреляций по Пирсону (мера линейной связи между признаками). Видно, что цена линейно коррелирует с объёмом двигателя. Стоит ожидать, что даже простые алгоритмы будут добиваться хороших метрик:

Для справки: взаимосвязь двух переменных проявляется в совместной вариации: при

```
1  import matplotlib.pyplot as plt
2
3  plt.subplots(figsize=(12, 10))
4  sns.heatmap(encoded_data.corr(), square = True, annot=True)
5  plt.show()
```



Закодируем теперь данные в one-hot формате. Найдём категориальные колонки и применим к ним функцию OneHotEncoder библиотеки Sklearn, которая переведёт их в бинарный формат. Выделим полученные названия новых колонок с помощью метода get_feature_names.

изменении одного показателя изменяется другой. Такая взаимосвязь называется корреляци ей

Сконкатинируем полученный one-hot датасет с остальными числовыми колонками.

```
# data_onehot = pd.get_dummies(df, columns=['model', 'transmission','fuelType'])

from sklearn.preprocessing import OneHotEncoder
# выделяем в DataFrame категориальные колонки (тип - object)
categor = df[['model', 'transmission','fuelType', 'year']]

# в случае, если появляется неизвестных новых - игнорируем
enc = OneHotEncoder(handle_unknown='ignore')
# выбираем какие на основе каких признаков производить преобразования
enc.fit(categor)
# производим преобразования категориальных
codes = enc.transform(categor).toarray()
# выделим названия категориальных колонок из тех, на основе которых преобразуем
feature_names = enc.get_feature_names(categor.columns)
# объединяем числовые и категориальные колонки

data_onehot = pd.concat([df[['price', 'mileage','tax', 'mpg', 'engineSize']], # исключаем категориальные - выбираем цифровые
                         pd.DataFrame(codes, columns = feature_names)], axis=1)# выбираем закодированные
```

Отобразим датасеты с label и one-hot кодированием. В случае one-hot у нас 219 колонок.

94324	1695	131000	200.0	39	1.9	0.0
94325	1450	147000	330.0	27	3.2	0.0

94326 rows × 219 columns

```
[ ]  1  df.shape
```

```
(94326, 9)
```

```
[ ]  1  data_onehot.reset_index(inplace=True, drop=True)
     2  data_onehot.shape
```

```
(94326, 219)
```

Разделим полученные варианты кодирования на train/test в соотношении 80 на 20:

▾ Train-test split

Разделим на треин-тест для обоих методов кодировки

```
[ ]  1  from sklearn.model_selection import train_test_split
     2
     3  X_train, X_test, y_train, y_test = train_test_split(data_onehot.drop(columns=['price']), data_onehot['price'], test_size=0.2)
```

```
[ ]  1  X_train_, X_test_, y_train_, y_test_ = train_test_split(encoded_data.drop(['price'], axis=1), encoded_data['price'], test_size=0.2)
```

Алгоритм
Импортируем алгоритм решающего дерева, инициализируем и обучим его на части train, спрогнозируем на test. Оценим по метрикам качества r2 и медианной абсолютной ошибке.

```
1  from sklearn.tree import DecisionTreeRegressor
2  from sklearn.metrics import r2_score, median_absolute_error
3
4  alg = DecisionTreeRegressor()
5  alg.fit(X_train, y_train)
6  preds = alg.predict(X_test)
7
8  f'r2 = {r2_score(y_test, preds):.2f} MedAE = {median_absolute_error(y_test, preds):.2f}'
```

```
'r2 = 0.92 MedAE = 990.00'
```

С учётом того, что среднеквадратичный разброс цен равен около 10 000 долларов, медианное отклонение прогноза от факта в тысячу более чем приемлемо, что дополнительно подтверждает высокий коэффициент детерминации.

```
'r2 = 0.92 MedAE = 990.00'
```

```
[ ]  1  # стандартное отклонение на тр
     2  y_train.std()
```

```
9964.508299376843
```

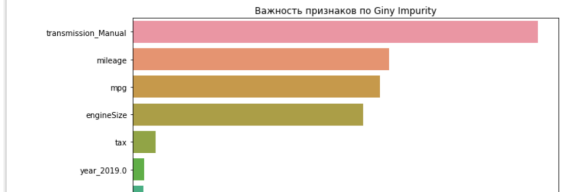
Стоит обратить внимание, что у алгоритма присутствует полезный аргумент feature_importances_, который рассчитывается с учетом положения признака в разделяющем дереве. Это позволяет оценить, какие признаки наиболее важны при разделении. Визуализируем их с найденными ранее названиями колонок. Видно, что такие характеристики как тип трансмиссии, пробег, расход топлива и объём двигателя наиболее важны в оценке цены.

```
[ ]  1  def plot_feature_importances(gs, column_names, top_n = 15):
     2      imp = pd.Series(gs.feature_importances_, index = column_names).sort_values(ascending=False)
     3      plt.figure(figsize=(10, 10))
     4      plt.title('Важность признаков по Giny Impurity')
     5      sns.barplot(x = imp.values[:top_n], y = imp.index.values[:top_n], orient='h')
```

```
[ ]  1  one_hot_feature_names[:5]
```

```
array(['price', 'mileage', 'tax', 'mpg', 'engineSize'], dtype=object)
```

```
1  plot_feature_importances(alg, one_hot_feature_names[1:])
```



<table>
<tr>
<td></td>
<td></td>
<td>Проведём аналогичный сценарий обучения с учетом one-hot кодирования. Качество отличается незначительно.


```
[ ]  1  from sklearn.tree import DecisionTreeRegressor
     2  from sklearn.metrics import r2_score, median_absolute_error
     3
     4  alg = DecisionTreeRegressor()
     5  alg.fit(X_train_, y_train_)
     6  preds = alg.predict(X_test_)
     7
     8  f'r2 = {r2_score(y_test_, preds):.2f} MedAE = {median_absolute_error(y_test_, preds):.2f}'

'r2 = 0.93 MedAE = 960.00'
```

</td>
<td></td>
</tr>
<tr>
<td>Закрепление изученного материала</td>
<td>15 мин.</td>
<td>Вопросы для обсуждения
● Из каких этапов состоит полный конвейер машинного обучения?
● С какими функциями библиотеки Sklearn мы познакомились?</td>
<td>Педагог организует беседу по вопросам</td>
</tr>
<tr>
<td>Этап подведения итогов занятия (рефлексия)</td>
<td>8 мин.</td>
<td>Вопросы для обсуждения
● Чему я научился?
● С какими трудностями я столкнулся?
● Какие вопросы остались? Что осталось непонятным?</td>
<td>Педагог способствует размышлению обучающихся над вопросами</td>
</tr>
</table>

Информация о домашнем задании, инструктаж по его применению	5 мин.	Дома повторите основные функции библиотек Pandas, Sklearn	

Рекомендуемые ресурсы для дополнительного изучения:

1. Введение в машинное обучение с помощью Scikit-learn. [Электронный ресурс] – Режим доступа: https://habr.com/ru/post/264241/
2. Библиотека Scikit-learn в Python. [Электронный ресурс] – Режим доступа: https://pythonim.ru/libraries/biblioteka-scikit-learn-v-python
3. Решающие деревья. [Электронный ресурс] – Режим доступа: https://ml-handbook.ru/chapters/decision_tree/intro