# ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

**Тема занятия:** Пайплайн выбора модели

**Аннотация к занятию:** на данном занятии обучающиеся познакомятся с библиотекой Sklearn. Обсудят, как можно перебирать различные гиперпараметры алгоритма.

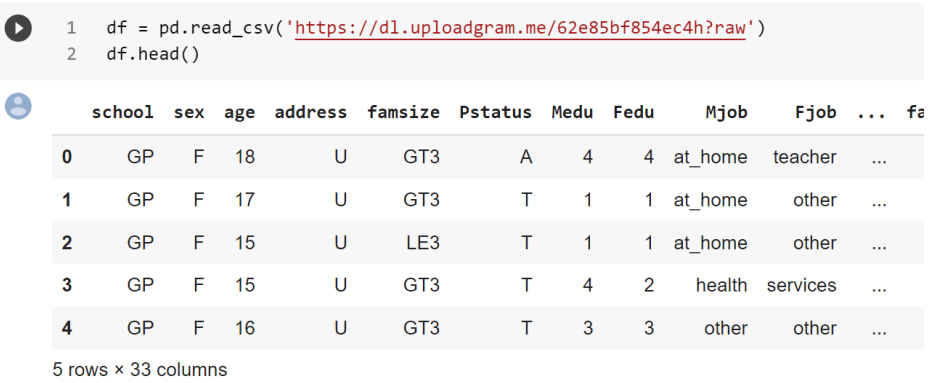
**Цель занятия:** сформировать у обучающихся представление о библиотеке Sklearn. Рассмотреть конвейер обработки данных с помощью различных моделей.

**Задачи занятия:**

- познакомить обучающихся с модулями grid search и pipeline библиотеки Sklearn;
- рассмотреть, как можно перебирать различные гиперпараметры алгоритма;
- применить полученные знания на практике.

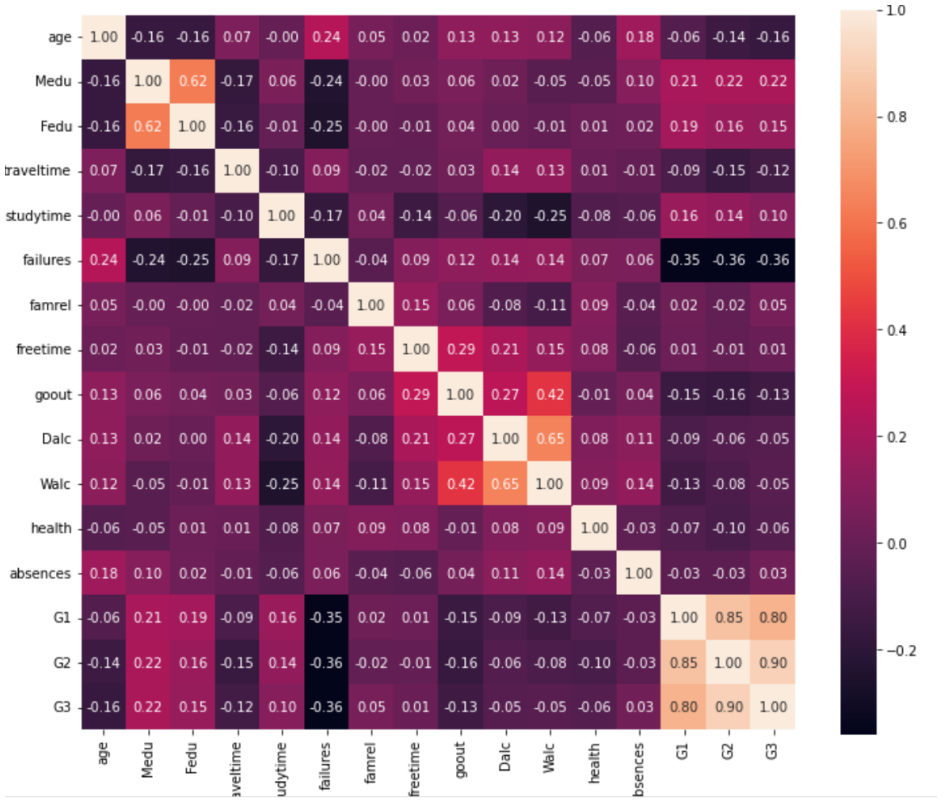
## Ход занятия

| Этап занятия | Время | Деятельность педагога | Комментарии, рекомендации для педагогов |
|---|---|---|---|
| **Организационный этап** | 2 мин. | Добрый день! Добро пожаловать на занятие! | Приветствие. Создание в классе атмосферы психологического комфорта |
| **Постановка цели и задач занятия. Мотивация учебной деятельности обучающихся** | 10 мин. | **Вопрос для обсуждения**<br>Как обучить алгоритм?<br><br>**Возможные ответы обучающихся**<br>На этом занятии мы обсудим, как можно перебирать различные гиперпараметры алгоритма. Иными словами, как автоматизировать поиск изначальных настроек алгоритма, наиболее подходящих для конкретных данных. На примере реальных данных мы построим конвейер, включающий исследование и перебор уже пройденных алгоритмов и их параметров. Выберем наиболее оптимальную по метрикам качества модель. | Способствовать обсуждению мотивационных вопросов |

<table>
<tr>
<td>Изучение нового материала</td>
<td>50 мин.</td>
<td>
В качестве данных мы используем датасет успеваемости учащихся по курсу математики в средней школе. Он состоит из 33 характеристик по типу пола, возраста, образования семьи и прочих. Подобные модели могут быть полезны для раннего выявления отстающих учеников.<br><br>
Загрузим данные по успеваемости. Для удобства они были скачаны с Kaggle и загружены на удалённый облачный сервер.<br><br>


<table>
<tr><th></th><th>school</th><th>sex</th><th>age</th><th>address</th><th>famsize</th><th>Pstatus</th><th>Medu</th><th>Fedu</th><th>Mjob</th><th>Fjob</th><th>...</th><th>fa</th></tr>
<tr><td>0</td><td>GP</td><td>F</td><td>18</td><td>U</td><td>GT3</td><td>A</td><td>4</td><td>4</td><td>at_home</td><td>teacher</td><td>...</td><td></td></tr>
<tr><td>1</td><td>GP</td><td>F</td><td>17</td><td>U</td><td>GT3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td></td></tr>
<tr><td>2</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>LE3</td><td>T</td><td>1</td><td>1</td><td>at_home</td><td>other</td><td>...</td><td></td></tr>
<tr><td>3</td><td>GP</td><td>F</td><td>15</td><td>U</td><td>GT3</td><td>T</td><td>4</td><td>2</td><td>health</td><td>services</td><td>...</td><td></td></tr>
<tr><td>4</td><td>GP</td><td>F</td><td>16</td><td>U</td><td>GT3</td><td>T</td><td>3</td><td>3</td><td>other</td><td>other</td><td>...</td><td></td></tr>
</table>
5 rows × 33 columns<br><br>
Отобразим типы данных. Нас интересует целевая переменная: в данном случае можно выбрать одну из трёх.
</td>
<td>
Для справки:<br>
Файл для работы:<br>
https://drive.google.com/file/d/1Y1bNZwulQeYE9hCWkSoiHy_R5gi4eelV/view?usp=sharing
</td>
</tr>
</table>

| | | | |
|---|---|---|---|
| | | health int64<br>absences int64<br>G1 int64<br>G2 int64<br>G3 int64<br>dtype: object<br><br>[ ]<br>```<br>df['G1'] = df['G1'].astype(int)<br>df['G2'] = df['G2'].astype(int)<br>df['G3'] = df['G3'].astype(int)<br>```<br><br>Рассмотрим коррелограмму — карту линейных корреляций признаков по Пирсону. Результаты всех трёх тестов (g1, g2, g3) схожи между собой, будем предсказывать g3. Явных линейных связей с целевой переменной от остальных признаков не наблюдается, что говорит нам о том, что простые линейные модели, вероятно, для этого не подходят. | Для справки: взаимосвязь двух переменных проявляется в совместной вариации: при изменении одного показателя меняется другой. Такая взаимосвязь называется коррел яцией. |




Подготовка данных

Выделим целевую переменную и разделим выборку на train и test в соотношении 80 на 20:

```
[ ]  1  from sklearn.model_selection import train_test_split
     2
     3  X_train, X_test, y_train, y_test = train_test_split(df.drop(columns = ['G1', 'G2', 'G3']), df['G3'], test_size=0.2, random_state=42)
```

```
[ ]  1  categorical_features = df.drop(columns=['G1', 'G2', 'G3']).select_dtypes(include= object).columns.values
     2  categorical_features
```

```
array(['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob',
       'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
       'nursery', 'higher', 'internet', 'romantic'], dtype=object)
```

Выделим столбцы по типу данных: категориальные и числовые.

```
1  categorical_features = df.drop(columns=['G1', 'G2', 'G3']).select_dtypes(include= object).columns.values
2  categorical_features
```

```
array(['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob',
       'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities',
       'nursery', 'higher', 'internet', 'romantic'], dtype=object)
```

```
1  numeric_features = df.drop(columns=['G1', 'G2', 'G3']).select_dtypes(exclude= object).columns.values
2  numeric_features
```

```
array(['age', 'Medu', 'Fedu', 'traveltime', 'studytime', 'failures',
       'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health',
       'absences'], dtype=object)
```

С последним обновлением библиотека Sklearn позволяет автоматизировать обработку колонок в зависимости от типа данных. Рассмотрим этот функционал:

```python
from sklearn.pipeline import make_pipeline # конвейер подготовки данных
from sklearn.pipeline import Pipeline # конвер алгоритмов
from sklearn.impute import SimpleImputer # пропуски заполняются медианными значениями
from sklearn.preprocessing import OneHotEncoder, StandardScaler # onehot encoding и стандартизации
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestRegressor
# конвейер подготовки данных числового типа
numeric_transformer = make_pipeline(SimpleImputer(strategy="median"),
                                    StandardScaler())

# конвейер подготовки данных категориального типа
preprocessor = ColumnTransformer(
    [
        ('num', numeric_transformer, numeric_features),
        (
            'cat',
            OneHotEncoder(handle_unknown='ignore', sparse=False),
            categorical_features
        )
    ],
    verbose_feature_names_out = False,
)
# подготовка и обучение алгоритма
alg = Pipeline(steps=[
    ('enc', preprocessor), # подготовка
    ('alg', RandomForestRegressor()) # алгоритм
])

# обучение алгоритма
alg.fit(X_train, y_train)
```

В первую очередь определим функцию для обработки колонок (column transformer). Для числовых (num) и категориальных (cat) колонок мы можем определить набор (названий) колонок и функции для их обработки. В нашем случае числовые колонки будет обрабатывать numeric transform, который включает функцию заполнения медианным значением пропусков и стандартизацию. Категориальные колонки же будут заполнены с помощью OneHotEncoder. Аргумент verbose_features_names_out функции ColumnTransformer будет добавлять ко всем именам признаков префикс имени преобразователя, сгенерировав этот признак при значении True. Нам это не нужно, установим его значение на False.

Теперь определим сам пайплайн, включающий себя как этап кодирования, описанный с помощью функции ColumnTransformer, так и сам алгоритм. В нашем случае будем использовать решающий лес (RandomForestRegressor). Удобство такой инициализации в том, что достаточно просто подставить вместо решающего леса любой другой алгоритм, представленный в Sklearn.

Обучение алгоритмов

Получим прогнозы на тесте и померяем метрики качества (коэффициент детерминации и медианную абсолютную ошибку). Видно, что пока алгоритм справляется относительно плохо.

```
# предсказания
preds = alg.predict(X_test)

# коэффициент детерминации и медианна абсолютная ошибка
r2_score(y_test, preds), median_absolute_error(y_test, preds)
```

```
(0.29472844450348523, 2.4000000000000004)
```

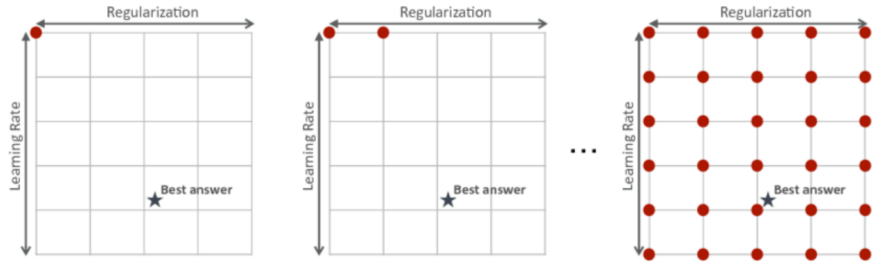
```
# среднее и стандартное отклонение целевой переменной
y_test.mean(), y_test.std()
```

```
(10.772151898734178, 4.55718471092093)
```

Попробуем задать сетку гиперпараметров алгоритма, которые мы можем перебрать для повышения его качества. Рассмотрим разное количество деревьев, минимальное количество примеров в листе и максимальное количество признаков.

Для справки: в задачах машинного обучения для оценки качества моделей и сравнения различных алгоритмов используются метрики.

```
[ ]  1  parameters = {
     2      # название этапа подготовки__гиперпараметр: [сетка (список) параметров]
     3      'alg__n_estimators': [100, 300, 500, 1000],
     4      'alg__min_samples_leaf': [1, 2, 4, 6],
     5      'alg__max_features': ['auto', 'sqrt', 'log2']
     6  }
```




Передадим алгоритм, сетку гиперпараметров, количество фолдов кросс-валидации и потоков для параллельной обработки в функцию GridSearchCV, которая найдёт нам наиболее оптимальные с точки зрения метрик качества параметры.

```
1  from sklearn.model_selection import GridSearchCV
2  # сетка гиперпараметров
3  gs = GridSearchCV(alg, # выбор алгоритма или пайплайна (подготовка + алгоритм)
4                    parameters, # сетка гиперпараметров
5                    cv=5, # количество фолдов кросс-валидации
6                    verbose=2, # отображать процесс перебора гиперпараметров
7                    n_jobs=2) # параллелизм вычислений
8
9  gs.fit(X_train, y_train)
10 gs.best_params_
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
{'alg__max_features': 'auto',
 'alg__min_samples_leaf': 1,
 'alg__n_estimators': 1000}
```

Используя лучший алгоритм, предскажем метрики на тестовой выборке. Ошибка незначительно изменилась в лучшую сторону. Целесообразно попробовать разные алгоритмы, подставив их вместо RandomForestRegressor) и поменяв соответствующую алгоритму сетку гиперпараметров.

Модуль кросс-валидации

```
[ ]  1  preds = gs.best_estimator_.predict(X_test)
     2
     3  r2_score(y_test, preds), median_absolute_error(y_test, preds)
```

```
(0.29946759506767107, 2.369999999999999)
```

| | | | |
|---|---|---|---|
| **Закрепление изученного материала** | 15 мин. | **Вопросы для обсуждения**<br>● Для чего нужна библиотека Sklearn?<br>● Как обучить алгоритм? | Педагог организует беседу по вопросам |
| **Этап подведения итогов занятия (рефлексия)** | 8 мин. | **Вопросы для обсуждения**<br>● Чему я научился?<br>● С какими трудностями я столкнулся?<br>● Какие вопросы остались? Что осталось непонятным? | Педагог способствует размышлению обучающихся над вопросами |
| **Информация о домашнем задании, инструктаж по его применению** | 5 мин. | Дома повторите основные функции библиотеки 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. От эскиза до релиза: пайплайн. [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/lightmap/blog/550968/