ТЕХНОЛОГИЧЕСКАЯ КАРТА ЗАНЯТИЯ

Тема занятия: Практика решающего леса

Аннотация к занятию: на данном занятии обучающиеся познакомятся с алгоритмом машинного обучения — случайным лесом. Рассмотрят на примере реальных данных качество алгоритма в зависимости от количества деревьев.

Цель занятия: сформировать у обучающихся представление об алгоритме машинного обучения — случайном лесе.

Задачи занятия:

- познакомить обучающихся с алгоритмом машинного обучения — случайным лесом;
- рассмотреть, как выглядят разделяющие поверхности в случае леса и одного дерева;
- 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.	**Вопрос для обсуждения** Как вы думаете, если в процессе прогнозирования участвует множество деревьев решений, данный метод будет точнее? **Ответы обучающихся** На этом занятии мы рассмотрим конвейер машинного обучения для алгоритма случайного леса. Мы загрузим и исследуем данные. На синтетическом примере визуализируем, как выглядят разделяющие поверхности в дереве и случайном лесу.	Способствовать обсуждению мотивационных вопросов
Изучение нового материала	50 мин.	Импортируем библиотеки, которые будут использованы в ходе анализа	Для справки: Файл для работы: https://drive.google.com/file/d/10VYvu1lkYtCuEFLtw

```python
import pandas as pd
from matplotlib.colors import ListedColormap
import numpy as np
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])
```

Сравнение дерева и случайного леса

Сравним, как происходит разделение классов деревом и решающим лесом. Для этого генерируем данные, которые использовались для теоретического занятия. Не будем в деталях останавливаться на механике реализации, её суть — создание точек 3 классов по 3 спиралям, закручивающимся к центру.

Код генерации датасета с занятия

Показать код

```python
X, y = devil(100)
```

Визуализируем эти данные, чтобы убедиться, что всё отработало корректно:

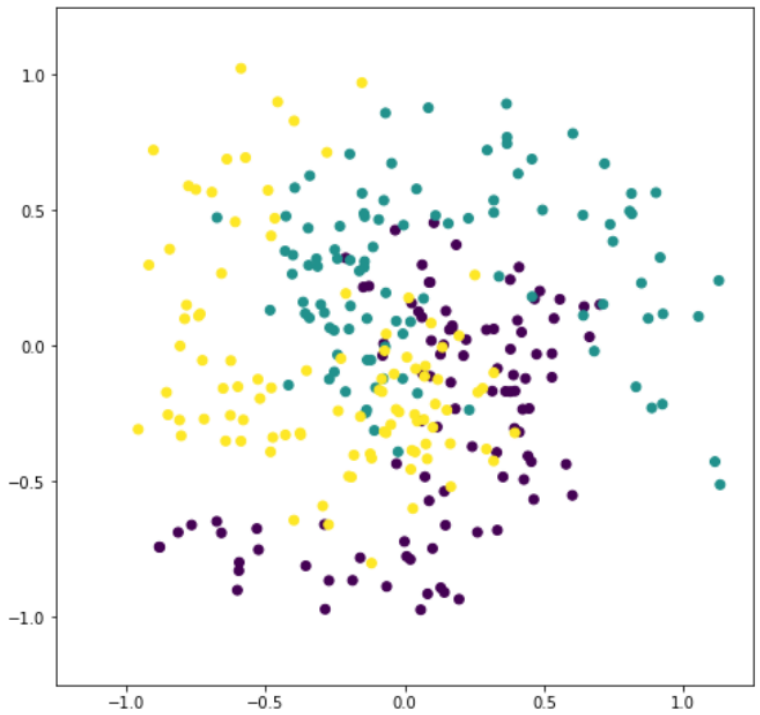
7G0NO2hT4b39oET/view?usp=sharing

```python
plt.figure(figsize=(8, 8))

plt.scatter(X[:,0], X[:,1], c=y)

plt.xlim([-1.25, 1.25])
plt.ylim([-1.25, 1.25])

plt.show()
```



Инициализируем две модели — решающего дерева и леса — с одинаковой глубиной.

```
models = [DecisionTreeClassifier(max_depth=10),
          RandomForestClassifier(max_depth=10)]
```

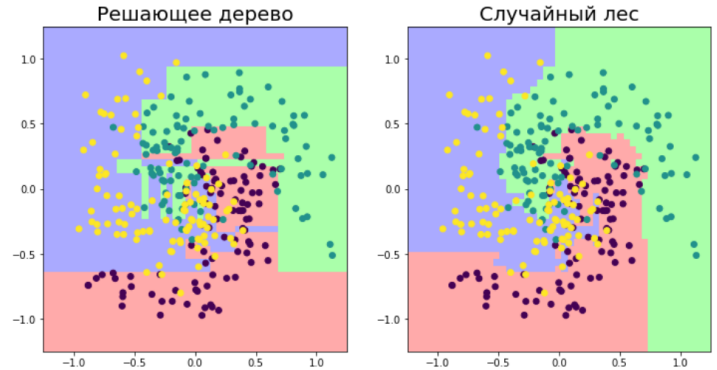
Создадим сетку, покрывающую диапазон данных по x, y. Обучим модели на созданных выше данных. Подскажем отношение класса к каждому элементу сетки. Визуализируем полученный прогноз для дерева и для леса. Видно, что границы разделяющего леса более плавные, что будет позитивно сказываться на метриках качества в дальнейшем.

```python
plt.figure(figsize=(12,6))
plt.title('Max depth = 7')

name = {0:'Решающее дерево', 1:'Случайный лес'}
for i, model in enumerate(models):
    model.fit(X, y)

    plt.subplot(1, 2, i+1)
    plt.xlim((-1.25, 1.25))
    plt.ylim((-1.25, 1.25))
    xx, yy = np.meshgrid(np.linspace(-1.25, 1.25, 50),
                         np.linspace(-1.25, 1.25, 50))
    plt.gca().set_title(name[i], fontsize=20)
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])

    Z = Z.reshape(xx.shape)
    plt.pcolormesh(xx, yy, Z, cmap=cmap_light)
    plt.scatter(X[:, 0], X[:, 1], c=y)
plt.show()
```



Данные

В ходе анализа данных будет использован датасет мобильных телефонов: https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?resource=download&select=train.csv

Подобные данные могут использоваться для создания автоматических подборок телефонов в зависимости от их характеристик ещё до анонса цены.

Наши данные были скачаны с платформы Kaggle и загружены на облачный сервер. Загрузим и проверим корректность данных. Видно, что все данные были загружены в свои колонки и явных сбоев с кодировкой нет.

```
df = pd.read_csv('https://dl.uploadgram.me/62def1f38f9cbh?raw')
```

```
df.head()
```

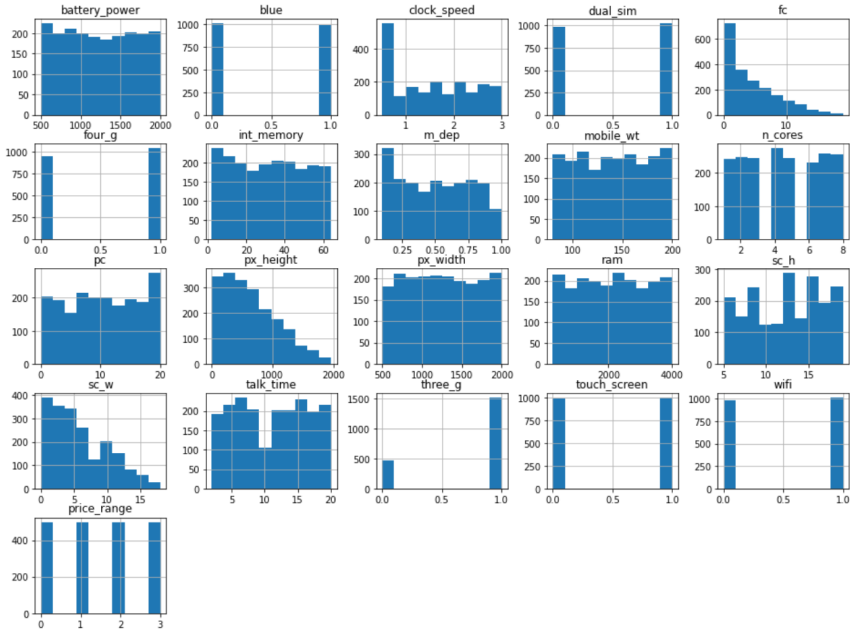
	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	m
0	842	0	2.2	0	1	0	7	0.6	
1	1021	1	0.5	1	0	1	53	0.7	
2	563	1	0.5	1	2	1	41	0.9	
3	615	1	2.5	0	0	0	10	0.8	
4	1821	1	1.2	0	13	1	44	0.6	

5 rows x 21 columns

Исследование

С помощью встроенного метода .hist визуализируем гистограммы по характеристикам телефонов. Можно видеть, что особенно аномальных значений нет.

```
dtype=object)
```

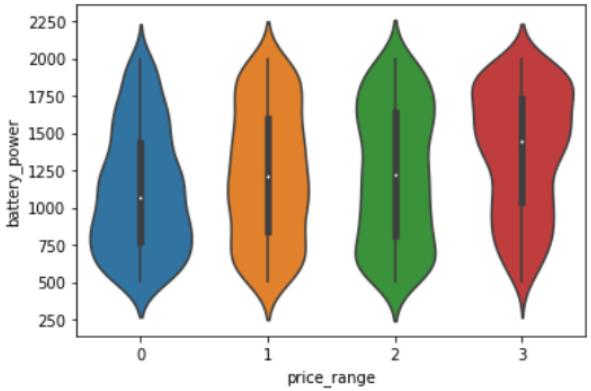


Помимо гистограмм, можно визуализировать связь характеристик с ценой. Например, медианный объем батареи растёт с ростом класса цены.

```
import seaborn as sns

sns.violinplot(x="price_range", y='battery_power', data=df)
```

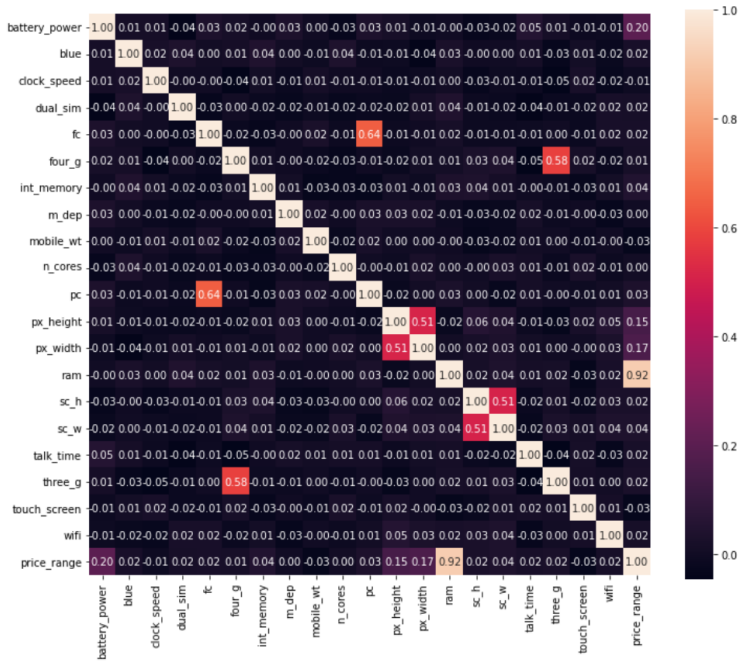
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9c3c609fd0>
```



Поскольку категориальных признаков в датасете нет, этапа кодирования тоже нет. Визуализируем гистограмму корреляций по Пирсону (мера линейной связи). Видно, например, что оперативная память линейно коррелирует с ценовой категорией.

```python
import matplotlib.pyplot as plt

plt.subplots(figsize=(12, 10))
sns.heatmap(df.corr(),fmt='.2f', square = True, annot=True)
plt.show()
```



Алгоритм

Разделим данные на train/test в соотношении 80 на 20.
Инициализируем алгоритм леса и обучим его на части train.
Проверим качество на test: усреднённый f1 в 0.86

показывает неплохие результаты алгоритма даже без перебора его параметров.

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

X_train, X_test, y_train, y_test = train_test_split(df.drop(columns = ['price_range']), df['price_range'], test_size=0.2)

# создание и обучение алгоритма
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
```

```
RandomForestClassifier()
```

```python
from sklearn.metrics import classification_report

preds = rf.predict(X_test)
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       0.92      0.93      0.92       105
           1       0.83      0.81      0.82       106
           2       0.78      0.81      0.80        96
           3       0.92      0.88      0.90        93

    accuracy                           0.86       400
   macro avg       0.86      0.86      0.86       400
weighted avg       0.86      0.86      0.86       400
```

Попробуем увеличить количество решающий деревьев и критерий разделения. Видим, что это даёт прирост в два процентных пункта.

```
rf = RandomForestClassifier(n_estimators=1000, criterion='entropy')
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=1000)
```

```
preds = rf.predict(X_test)
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       105
           1       0.86      0.87      0.86       106
           2       0.82      0.81      0.82        96
           3       0.91      0.89      0.90        93

    accuracy                           0.88       400
   macro avg       0.88      0.88      0.88       400
weighted avg       0.88      0.88      0.88       400
```

Для наглядности рассмотрим несколько примеров из тестовой части и спрогнозируем их. Видно, что прогнозы полностью совпадают с разметкой.

```
[ ]  1  X_test.iloc[:5]
```

	battery_power	blue	clock_speed	dual_si
1972	1191	0	0.8	(
859	623	0	2.0	(
384	625	1	1.9	(
1053	623	1	0.9	(
297	1329	1	1.0	(

```
1  print('Предсказания:')
2  print(rf.predict(X_test.iloc[0:5,].values)
3  print('Реальные ответ:')
4  y_test[:5].values
```

```
Предсказания:
[0 2 3 3 1]
Реальные ответ:
/usr/local/lib/python3.7/dist-packages/sklearn/b
  "X does not have valid feature names, but"
array([0, 2, 2, 3, 1])
```

Сравним с алгоритмом логистической регрессии и решающего дерева. В обоих случаях результаты макро-f1 хуже.

```python
from sklearn.linear_model import LogisticRegression

tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)

lr = LogisticRegression()
lr.fit(X_train, y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the d
    https://scikit-learn.org/stable/modules/preprocessing.h
Please also refer to the documentation for alternative solv
    https://scikit-learn.org/stable/modules/linear_model.ht
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

```python
preds = tree.predict(X_test)
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       105
           1       0.80      0.85      0.83       106
           2       0.80      0.78      0.79        96
           3       0.90      0.86      0.88        93

    accuracy                           0.85       400
   macro avg       0.85      0.85      0.85       400
weighted avg       0.85      0.85      0.85       400
```

<table>
<tr>
<td></td>
<td></td>
<td>

```
preds = lr.predict(X_test)
print(classification_report(y_test, preds))
```

```
              precision    recall  f1-score   support

           0       0.84      0.74      0.79       105
           1       0.57      0.62      0.60       106
           2       0.52      0.46      0.49        96
           3       0.68      0.78      0.73        93

    accuracy                           0.65       400
   macro avg       0.65      0.65      0.65       400
weighted avg       0.66      0.65      0.65       400
```

</td>
<td></td>
</tr>
<tr>
<td>Закрепление изученного материала</td>
<td>15 мин.</td>
<td>Вопросы для обсуждения
● В чём заключается смысл алгоритма случайного леса?
● При каком алгоритме получается более точный результат?</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.	-	

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
4. Метод случайного леса. [Электронный ресурс] – Режим доступа: https://habr.com/ru/company/ods/blog/324402/