Определение возраста покупателей по фотографии

Клиентом поставлена задача - определить возраст покупателя по фотографии. Этот механизм будет применяться для контроля продажи товаров с ограничениями по возрасту (сигареты, алкоголь), а также для анализа покупателей в различных торговых точках для улучшения таргетинга рекламных акций.

Для обучения модели возьмем набор данных с ресурса https://chalearnlap.cvc.uab.es/dataset/26/description/, на котором есть требуемая нам разметка - фотографии с определенным возрастом.

Исследовательский анализ данных

Загружаем необходимые библиотеки

```
from tensorflow import keras


from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, Flatten, Dense, AvgPool2D, GlobalAveragePooling2D
import numpy as np
from keras.optimizers import Adam
from keras.applications.resnet import ResNet50
import pandas as pd
import matplotlib.pyplot as plt
import shutil
```

Загрузим данные

```
!gdown --id 1nJmj86j770ULoprRwXUGcnmyaWB5FMqb
```

```
/usr/local/lib/python3.10/dist-packages/gdown/cli.py:121: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be r
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1nJmj86j770ULoprRwXUGcnmyaWB5FMqb
To: /content/face_age.zip
100% 239M/239M [00:01<00:00, 225MB/s]
```

```
shutil.unpack_archive('/content/face_age.zip')
```

```
labels = pd.read_csv('/content/labels.csv')
```

```
labels.head()
```

	file_name	real_age
0	000000.jpg	4
1	000001.jpg	18
2	000002.jpg	80
3	000003.jpg	50
4	000004.jpg	17

```
train_datagen = ImageDataGenerator(rescale=1./255)
train_gen_flow = train_datagen.flow_from_dataframe(
        dataframe=labels,
        directory='/content/final_files/',
        x_col='file_name',
        y_col='real_age',
        target_size=(224, 224),
        batch_size=32,
        class_mode='raw',
        seed=12345)
```

```
Found 7591 validated image filenames.
```

Выборка состоит из 7591 фотографий

```
labels.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7591 entries, 0 to 7590
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   file_name  7591 non-null   object
 1   real_age   7591 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 118.7+ KB
```

С данными все в порядке, пропущенных значений нет.

```
labels.head()
```

	file_name	real_age
0	000000.jpg	4
1	000001.jpg	18
2	000002.jpg	80
3	000003.jpg	50
4	000004.jpg	17

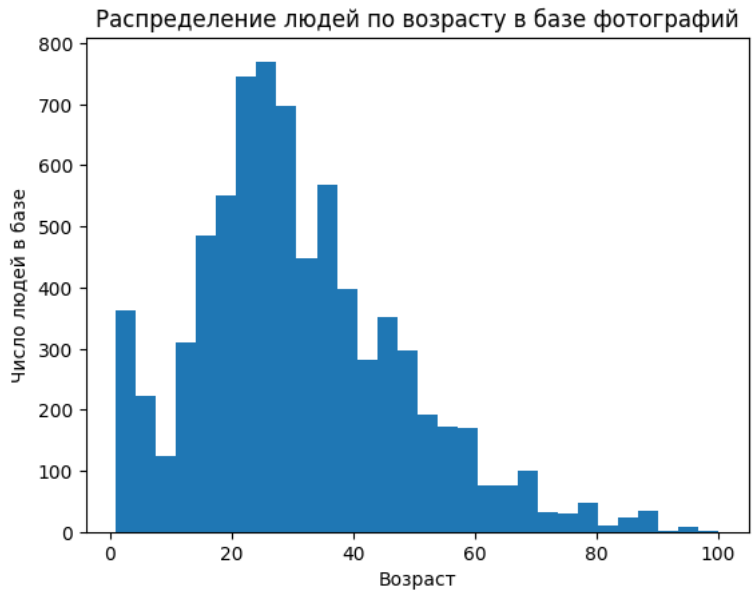
```
labels['real_age'].describe()
```

```
count    7591.000000
mean       31.201159
std        17.145060
min         1.000000
25%        20.000000
50%        29.000000
75%        41.000000
max       100.000000
Name: real_age, dtype: float64
```

Минимальный возраст покупателя - 1 год. С одной стороны, конечно, покупателю с возрастом в 1 год вряд ли что то можно предложить. С другой стороны, его родителям - вполне. Поэтому не стоит ограничивать выборку снизу по возрасту.

Посмотрим на распределение

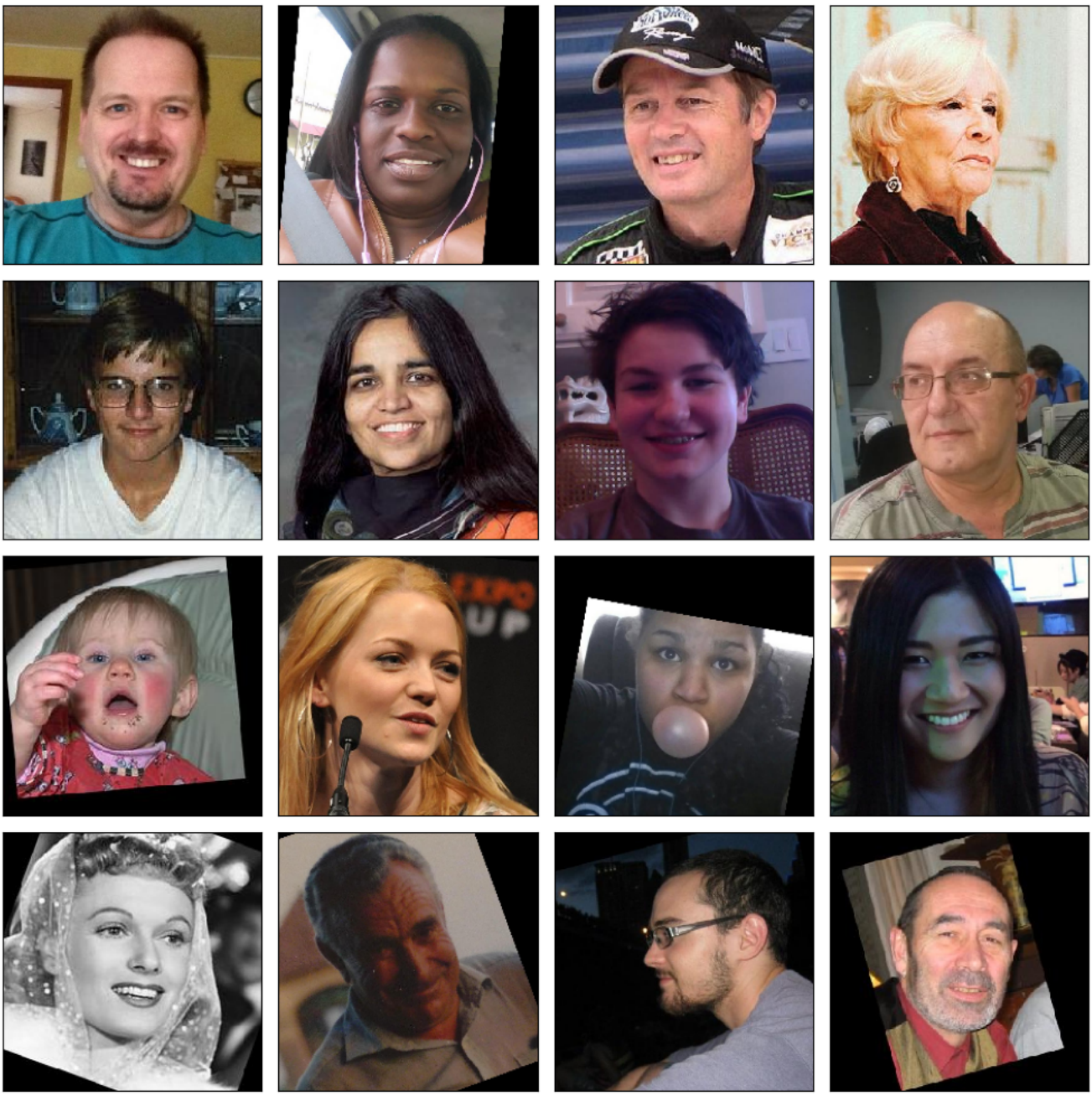
```
fig = plt.figure()
plt.hist(labels['real_age'], 30)
plt.xlabel('Возраст')
plt.ylabel('Число людей в базе')
plt.title('Распределение людей по возрасту в базе фотографий');
```



Наблюдается провал в возрасте примерно 10 лет. До этого возраста дети обычно в магазин приходят с родителями, а похже постепенно начинают ходить уже сами. Провал наблюдается видимо из-за того, что совсем маленьких детей меньше оставляют дома одних - и они тоже "приходят" в магазин. А начиная лес с 5-6 они уже могут и дома остаться. И в районе примерно 9 лет наблюдается минимум как раз из-за того, что они уже чаще всего остаются дома и не ходят с родителями, а сами еще по большей части ходить по магазинам не начали.

```
features, target = next(train_gen_flow)

# выводим 16 изображений
fig = plt.figure(figsize=(10,10))
for i in range(16):
    fig.add_subplot(4, 4, i+1)
    plt.imshow(features[i])
    # для компактности удаляем оси и прижимаем изображения друг к другу
    plt.xticks([])
    plt.yticks([])
    plt.tight_layout()
```



```
features.shape
```

```
(32, 224, 224, 3)
```

Размер фотографий 224*224

Выводы по базе фотографий

Размер имеющихся фотографий: 224*224 пиксела, полноцветные (3 канала)

Возраст людей на имеющихся фото распределен от 1 до 100 лет с максимумом около 25-30 лет.

Наблюдается провал в возрасте примерно 10 лет. До этого возраста дети обычно в магазин приходят с родителями, а позже постепенно начинают ходить уже сами. Провал наблюдается видимо из-за того, что совсем маленьких детей меньше оставляют дома одних - и они тоже "приходят" в магазин. А начиная лес с 5-6 они уже могут и дома остаться. И в районе примерно 9 лет наблюдается минимум как раз из-за того, что они уже чаще всего остаются дома и не ходят с родителями, а сами еще по большей части ходить по магазинам не начали.

Вряд ли в магазине удастся сфотографировать на камеру портрет грудного ребенка, но исключить это тоже нельзя, поэтому мы не будем убирать эти данные из выборки

Фотографии частично повернуты так, чтобы лицо было вертикально. При обучении можно еще сделать вариант аугментации с отражением по горизонтали. Также могут помочь небольшие сдвиги и повороты

Обучение модели

```
def load_train(path):
    labels = pd.read_csv(path + 'labels.csv')
    index_train = int(len(labels) * 0.75)

    train_datagen = ImageDataGenerator(
        rescale=1./255,
        horizontal_flip=True,
        rotation_range=20,
        width_shift_range = 0.2,
        height_shift_range=0.2
        #vertical_flip=True
        )

    train_datagen_flow = train_datagen.flow_from_dataframe(
            dataframe=labels[:index_train],
            directory= path + 'final_files/',
            x_col='file_name',
            y_col='real_age',
            target_size=(224, 224),
            batch_size=32,
            class_mode='raw',
            seed=12345)

    return train_datagen_flow

def load_test(path):

    labels = pd.read_csv(path + 'labels.csv')
    index_train = int(len(labels) * 0.75)

    test_datagen = ImageDataGenerator(
        rescale=1./255,
        #horizontal_flip=True,
        #rotation_range=20,
        #width_shift_range = 0.2,
        #height_shift_range=0.2
        )

    test_datagen_flow = test_datagen.flow_from_dataframe(
            dataframe=labels[index_train:],
            directory= path + 'final_files/',
            x_col='file_name',
            y_col='real_age',
            target_size=(224, 224),
            batch_size=32,
            class_mode='raw',
            seed=12345)

    return test_datagen_flow

def create_model(input_shape=(224, 224, 3)):
    backbone = ResNet50(input_shape=input_shape,
                        #weights='/content/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5',
                        #by_name=True,
                        weights='imagenet',
                        include_top=False)
```

```
    # замораживаем ResNet50 без верхушки
    backbone.trainable = True

    model = Sequential()
    model.add(backbone)
    model.add(GlobalAveragePooling2D())
    #model.add(Dense(100, activation='relu'))
    model.add(Dense(1))

    optimizer = Adam(learning_rate=0.00005)

    model.compile(loss='mean_squared_error', optimizer=optimizer, metrics=['mae'])

    return model

def train_model(model, train_data, test_data, batch_size=None, epochs=20,
                steps_per_epoch=None, validation_steps=None):

    if steps_per_epoch is None:
        steps_per_epoch = len(train_data)
    if validation_steps is None:
        validation_steps = len(test_data)

    model.fit(train_data,
              validation_data=test_data,
              batch_size=batch_size, epochs=epochs,
              steps_per_epoch=steps_per_epoch,
              validation_steps=validation_steps,
              verbose=2)
    return model


path = '/content/'


train = load_train(path)
test = load_test(path)
model = create_model()
model_trained = train_model(model, train, test, batch_size=32, epochs=40)
```

```
Found 5693 validated image filenames.
Found 1898 validated image filenames.
Epoch 1/40
178/178 - 149s - loss: 288.3410 - mae: 12.3849 - val_loss: 1087.0880 - val_mae: 27.2535 - 149s/epoch - 839ms/step
Epoch 2/40
178/178 - 109s - loss: 90.2701 - mae: 7.1886 - val_loss: 1103.6832 - val_mae: 27.5055 - 109s/epoch - 611ms/step
Epoch 3/40
178/178 - 110s - loss: 70.0163 - mae: 6.4016 - val_loss: 740.5122 - val_mae: 20.8371 - 110s/epoch - 619ms/step
Epoch 4/40
178/178 - 114s - loss: 63.3472 - mae: 6.0549 - val_loss: 354.4753 - val_mae: 14.1178 - 114s/epoch - 638ms/step
Epoch 5/40
178/178 - 115s - loss: 53.2391 - mae: 5.5943 - val_loss: 193.4529 - val_mae: 10.3756 - 115s/epoch - 645ms/step
Epoch 6/40
178/178 - 112s - loss: 47.3557 - mae: 5.2927 - val_loss: 122.3467 - val_mae: 7.9369 - 112s/epoch - 631ms/step
Epoch 7/40
178/178 - 114s - loss: 44.4271 - mae: 5.1217 - val_loss: 190.5443 - val_mae: 9.9844 - 114s/epoch - 639ms/step
Epoch 8/40
178/178 - 109s - loss: 39.0666 - mae: 4.8265 - val_loss: 110.9936 - val_mae: 7.7137 - 109s/epoch - 613ms/step
Epoch 9/40
178/178 - 112s - loss: 37.2540 - mae: 4.7006 - val_loss: 103.4046 - val_mae: 7.3344 - 112s/epoch - 627ms/step
Epoch 10/40
178/178 - 112s - loss: 34.0258 - mae: 4.4602 - val_loss: 116.8582 - val_mae: 7.5411 - 112s/epoch - 631ms/step
Epoch 11/40
178/178 - 112s - loss: 32.6703 - mae: 4.3811 - val_loss: 128.7580 - val_mae: 7.8888 - 112s/epoch - 627ms/step
Epoch 12/40
178/178 - 107s - loss: 28.9895 - mae: 4.1722 - val_loss: 100.9676 - val_mae: 7.3924 - 107s/epoch - 602ms/step
Epoch 13/40
178/178 - 109s - loss: 26.8368 - mae: 3.9911 - val_loss: 170.0202 - val_mae: 9.2520 - 109s/epoch - 611ms/step
Epoch 14/40
178/178 - 110s - loss: 25.7874 - mae: 3.8797 - val_loss: 134.8931 - val_mae: 9.0606 - 110s/epoch - 618ms/step
Epoch 15/40
178/178 - 111s - loss: 25.5925 - mae: 3.8991 - val_loss: 99.5627 - val_mae: 7.2270 - 111s/epoch - 626ms/step
Epoch 16/40
178/178 - 110s - loss: 22.4270 - mae: 3.6510 - val_loss: 113.2789 - val_mae: 7.6045 - 110s/epoch - 616ms/step
Epoch 17/40
178/178 - 109s - loss: 22.4484 - mae: 3.6702 - val_loss: 102.0999 - val_mae: 7.1787 - 109s/epoch - 611ms/step
Epoch 18/40
178/178 - 109s - loss: 21.3732 - mae: 3.5382 - val_loss: 116.4659 - val_mae: 7.6867 - 109s/epoch - 614ms/step
Epoch 19/40
178/178 - 108s - loss: 20.9085 - mae: 3.5185 - val_loss: 163.6395 - val_mae: 9.4039 - 108s/epoch - 609ms/step
Epoch 20/40
178/178 - 109s - loss: 17.8912 - mae: 3.2654 - val_loss: 100.4861 - val_mae: 7.1210 - 109s/epoch - 613ms/step
Epoch 21/40
178/178 - 115s - loss: 17.4353 - mae: 3.2510 - val_loss: 99.0985 - val_mae: 7.1157 - 115s/epoch - 646ms/step
Epoch 22/40
178/178 - 108s - loss: 16.2617 - mae: 3.0895 - val_loss: 101.9127 - val_mae: 7.2058 - 108s/epoch - 607ms/step
```

```
Epoch 23/40
178/178 - 110s - loss: 17.2762 - mae: 3.1942 - val_loss: 121.3112 - val_mae: 7.7547 - 110s/epoch - 617ms/step
Epoch 24/40
178/178 - 108s - loss: 16.4580 - mae: 3.1038 - val_loss: 96.5372 - val_mae: 7.0656 - 108s/epoch - 607ms/step
Epoch 25/40
178/178 - 115s - loss: 15.1115 - mae: 3.0100 - val_loss: 134.5678 - val_mae: 8.0638 - 115s/epoch - 645ms/step
Epoch 26/40
178/178 - 112s - loss: 15.9161 - mae: 3.0648 - val_loss: 102.3655 - val_mae: 7.1400 - 112s/epoch - 631ms/step
Epoch 27/40
178/178 - 109s - loss: 13.0885 - mae: 2.7889 - val_loss: 101.5990 - val_mae: 7.0478 - 109s/epoch - 614ms/step
Epoch 28/40
```

Вывод обучения модели:

Окончательный MAE на валидационной выборке: ~7.0. На этот уровень обучение вышло примерно к 30 эпохе и далее результат не улучшался

Анализ обученной модели

Исследовательский анализ данных показал, что фотографии в имеющемся банке имеют размер 244*244 пиксела, в трех цветовых каналах (RGB)

Проведено обучение нейронной сети с использованием ResNet50 и одним нейроном на выходе.

Результат обученной модели имеет MAE на валидационной выборке 7.0. Это позволяет с достаточно высокой точностью определить возраст человека. Правда, конечно, пограничный возраст определить сложно. То есть, если, к примеру, продавец продаст сигареты или алкоголь 12-летнему ребенку, это будет поймано сразу. А вот проверять, что там с возрастом 16-20 лет нужно отдельно.

Однако модель можно успешно использовать для рекомендательных систем. Ошибка в 7 лет позволяет хорошо сегментировать покупателей.

Обучение было проведено с переобучением всех весов ResNet50. Отдельно был протестирован вариант с обучением только последнего слоя. Но в этом случае результат получился хуже (MAE=~12). То есть наша выборка фотографий лиц оказалась достаточно специфичной, что потребовало переобучения всех весов модели.