Определение возраста покупателей

Сетевой супермаркет внедряет систему компьютерного зрения для обработки фотографий покупателей. Фотофиксация в прикассовой зоне поможет определять возраст клиентов, чтобы:

- Анализировать покупки и предлагать товары, которые могут заинтересовать покупателей этой возрастной группы;
- Контролировать добросовестность кассиров при продаже алкоголя.

Цель исследования — построить модель, которая по фотографии определит приблизительный возраст человека.

В нашем распоряжении набор фотографий людей с указанием возраста. Данные взяты с сайта ChaLearn Looking at People.

Исследование пройдёт в три этапа:

- Исследовательский анализ данных
- Обучение модели
- Анализ обученной модели

Исследовательский анализ данных

In [1]:
```
# Импорт необходимых библиотек


import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet import ResNet50
```

In [2]:
```
# Константы


r_seed = 12345
```

In [3]:
```
# Функции и классы


# Функция для получения общих сведений о данных

def data_info(title, data):

    print('Общие сведения "{}":'.format(title))
    print()

    data.info()

    print()
    print()

    print('Пример данных (случайные 5 строк):')
    display(data.sample(5, random_state=r_seed))

    print()
    print()

    print('Количество пропусков по столбцам:')
    print()
    for col in data.columns:
        nmv = data[col].isna().sum()
        pmv = nmv/len(data)

        if pmv == 0:
            print('\033[0m{} - {} шт. - {:.2%}'.format(col, nmv, pmv))
        elif pmv <= 0.1:
            print('\033[0m{} - \033[43m{} шт.\033[0m - \033[43m{:.2%}'.format(col, nmv, pmv))
        else:
            print('\033[0m{} - \033[41m{} шт.\033[0m - \033[41m{:.2%}'.format(col, nmv, pmv))
        print('\033[0m')

    print()

    print('Количество уникальных значений в столбцах:')
    print()
    for col in data.columns:
```

```
        print('{} - {}'.format(col, data[col].nunique()))

    print()

    print('Описательная статистика данных:')
    print()
    print(data.describe())

    print()
    print()
```

Загрузим данные

In [4]:
```
labels = pd.read_csv('/datasets/faces/labels.csv')

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

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

In [5]:
```
data_info('labels', labels)
```

```
Общие сведения "labels":

<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


Пример данных (случайные 5 строк):
```

	file_name	real_age
5370	005370.jpg	50
4516	004516.jpg	41
3968	003968.jpg	46
2188	002188.jpg	85
5833	005833.jpg	14

```
Количество пропусков по столбцам:

file_name - 0 шт. - 0.00%

real_age - 0 шт. - 0.00%


Количество уникальных значений в столбцах:

file_name - 7591
real_age - 97

Описательная статистика данных:

          real_age
count  7591.000000
mean     31.201159
std      17.145060
min       1.000000
25%      20.000000
50%      29.000000
75%      41.000000
max     100.000000
```

Описание данных:

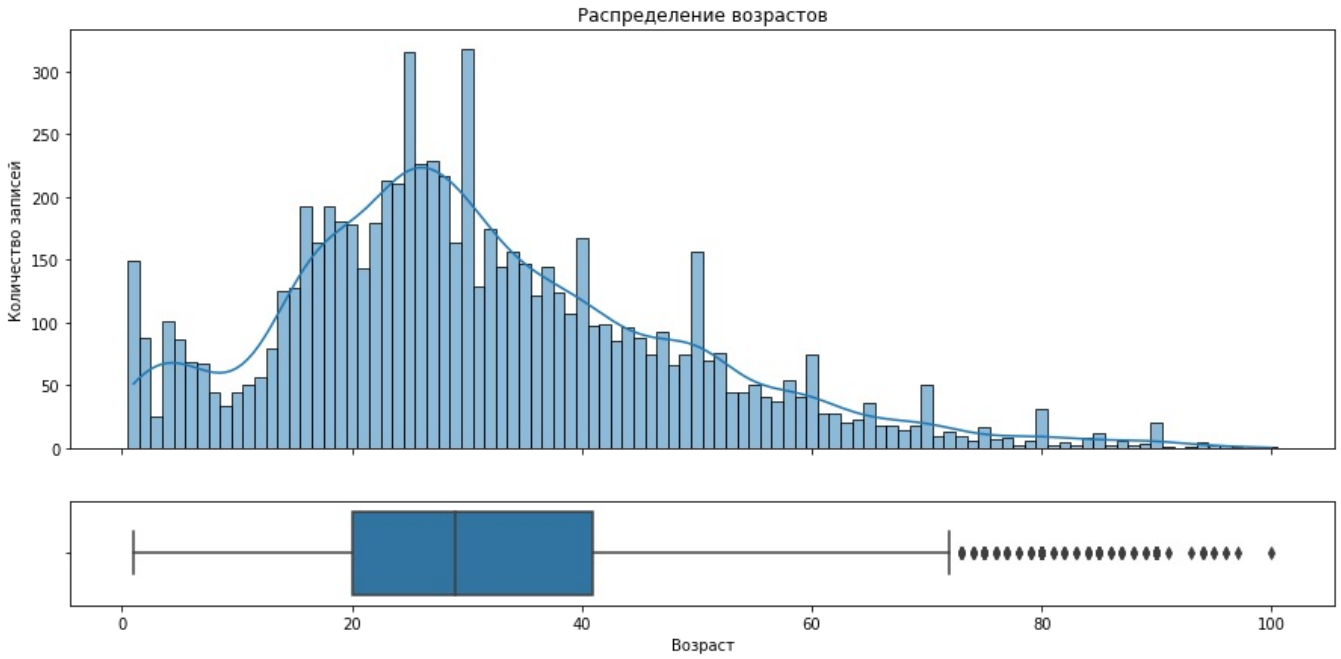
- Данные состоят из 7591 объектов
- Labels.csv имеет 2 столбца: Наименование фото объекта и его реальный возраст
- Пропуски отсутствуют

- Средний возраст 31 год
- Минимальный возраст 1 год
- Максимальный возраст 100 лет

In [6]:
```
_, [axs_hist, axs_box] = plt.subplots(2, 1,
                                      figsize=(15, 7),
                                      sharex=True,
                                      gridspec_kw=dict(height_ratios=[4,1]))

sns.histplot(data=labels,
             x='real_age',
             bins=labels['real_age'].max(),
             discrete=True,
             kde=True,
             ax=axs_hist)\
.set(title='Распределение возрастов', ylabel='Количество записей')

sns.boxplot(data=labels, x='real_age', ax=axs_box)\
.set(xlabel='Возраст');
```

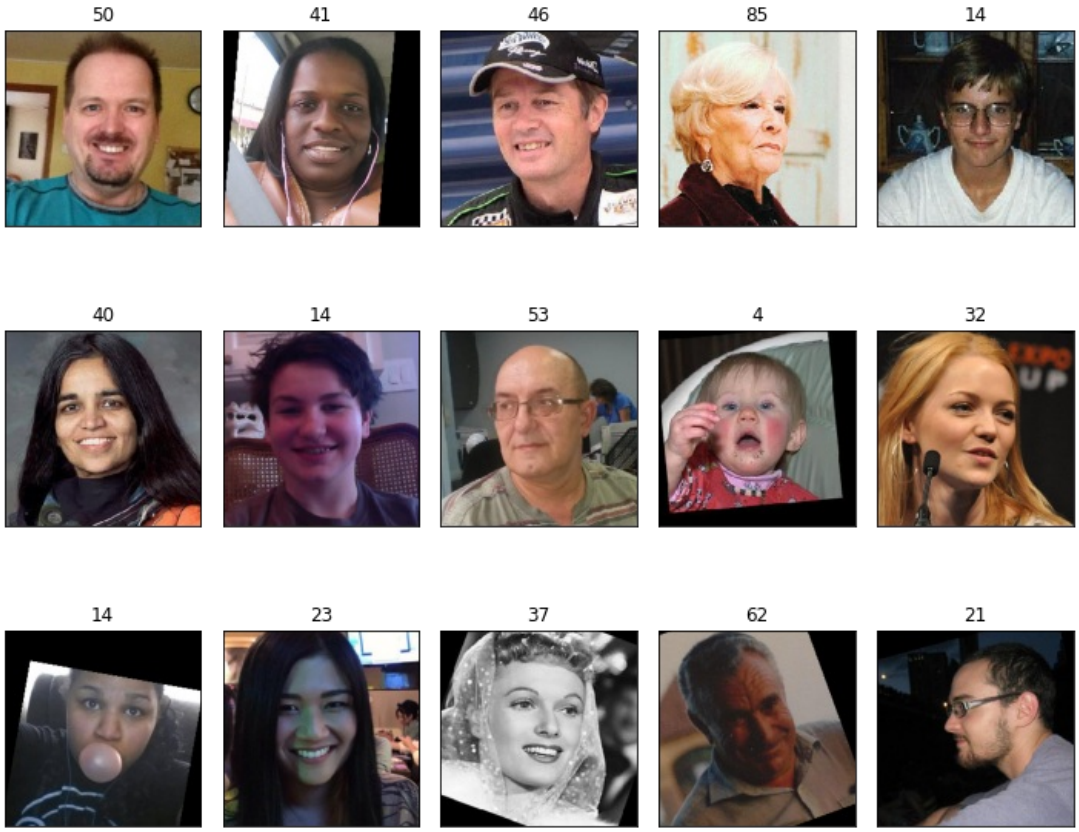


In [7]:
```
labels['real_age'].value_counts(ascending=False).head(20)
```

Out[7]:
```
30    317
25    315
27    229
26    226
28    217
23    213
24    211
18    193
16    193
19    180
22    179
20    178
32    174
40    167
29    164
17    163
34    156
50    156
1     149
35    147
Name: real_age, dtype: int64
```

In [8]:
```
features, target = next(train_gen_flow)

fig = plt.figure(figsize=(10,10))
for i in range(15):
    fig.add_subplot(3, 5, i+1)
    plt.imshow(features[i])
    plt.title(target[i])
    plt.xticks([])
    plt.yticks([])
    plt.tight_layout()
```



Вывод:

- Всего 7591 объектов
- Большинство людей в выборке находятся в возрасте от 20 до 40 лет
- В данных присутствуют черно-белые фото и фото под углом

Обучение модели

(Код в этом разделе запускался в отдельном GPU-тренажёре, поэтому оформлен не как ячейка с кодом, а как код в текстовой ячейке)

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, Flatten, Dense, AvgPool2D, GlobalAveragePooling2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.resnet import ResNet50
import pandas as pd

datagen = ImageDataGenerator(validation_split=0.25, rescale=1./255)
r_seed = 12345

def load_train(path):
    dataframe = pd.read_csv(path + '/labels.csv')
    directory = path +'/final_files'
    train_gen_flow = datagen.flow_from_dataframe(
        dataframe=dataframe,
        directory=directory,
        x_col='file_name',
        y_col='real_age',
        target_size=(224, 224),
        batch_size=32,
        class_mode='raw',
        subset='training',
        seed=r_seed)
    return train_gen_flow

def load_test(path):
    dataframe = pd.read_csv(path + '/labels.csv')
    directory = path +'/final_files'
    test_gen_flow = datagen.flow_from_dataframe(
        dataframe=dataframe,
        directory=directory,
        x_col='file_name',
        y_col='real_age',
        target_size=(224, 224),
```

```
        batch_size=32,
        class_mode='raw',
        subset='validation',
        seed=r_seed)
    return test_gen_flow

def create_model(input_shape):
    backbone = ResNet50(input_shape=input_shape,
                    weights='imagenet',
                    include_top=False)

    model = Sequential()
    optimizer = Adam(lr=0.0001)

    model.add(backbone)
    model.add(GlobalAveragePooling2D())
    model.add(Dense(1, activation='relu'))
    model.compile(loss='mean_absolute_error', optimizer=optimizer, metrics=['mean_absolute_error'])

    return model


def train_model(model, train_data, test_data, batch_size=None, epochs=16,
                steps_per_epoch=None, validation_steps=None):
    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
```

```
Train for 178 steps, validate for 60 steps
Epoch 1/16
2023-09-13 18:02:58.111626: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
Successfully opened dynamic library libcublas.so.10
2023-09-13 18:02:59.272464: I tensorflow/stream_executor/platform/default/dso_loader.cc:44]
Successfully opened dynamic library libcudnn.so.7
178/178 - 73s - loss: 11.5904 - mean_absolute_error: 11.5914 - val_loss: 19.9552 -
val_mean_absolute_error: 19.9472
Epoch 2/16
178/178 - 39s - loss: 6.7388 - mean_absolute_error: 6.7393 - val_loss: 22.9074 -
val_mean_absolute_error: 22.9082
Epoch 3/16
178/178 - 41s - loss: 5.5765 - mean_absolute_error: 5.5766 - val_loss: 20.0554 -
val_mean_absolute_error: 20.0546
Epoch 4/16
178/178 - 42s - loss: 4.8079 - mean_absolute_error: 4.8083 - val_loss: 10.9997 -
val_mean_absolute_error: 11.0175
Epoch 5/16
178/178 - 56s - loss: 4.2071 - mean_absolute_error: 4.2071 - val_loss: 6.7737 -
val_mean_absolute_error: 6.7880
Epoch 6/16
178/178 - 59s - loss: 3.8090 - mean_absolute_error: 3.8095 - val_loss: 6.8266 -
val_mean_absolute_error: 6.8308
Epoch 7/16
178/178 - 42s - loss: 3.4915 - mean_absolute_error: 3.4917 - val_loss: 6.2957 -
val_mean_absolute_error: 6.3018
Epoch 8/16
Epoch 9/16
178/178 - 50s - loss: 3.1720 - mean_absolute_error: 3.1721 - val_loss: 5.8484 -
val_mean_absolute_error: 5.8732
178/178 - 62s - loss: 2.9201 - mean_absolute_error: 2.9203 - val_loss: 6.7034 -
val_mean_absolute_error: 6.7045
Epoch 10/16
178/178 - 51s - loss: 2.8152 - mean_absolute_error: 2.8150 - val_loss: 5.9998 -
val_mean_absolute_error: 6.0135
Epoch 11/16
178/178 - 54s - loss: 2.5353 - mean_absolute_error: 2.5354 - val_loss: 6.1986 -
val_mean_absolute_error: 6.2143
Epoch 12/16
178/178 - 61s - loss: 2.4797 - mean_absolute_error: 2.4794 - val_loss: 6.2355 -
val_mean_absolute_error: 6.2332
Epoch 13/16
178/178 - 45s - loss: 2.3920 - mean_absolute_error: 2.3921 - val_loss: 6.1225 -
val_mean_absolute_error: 6.1374
```

```
Epoch 14/16
178/178 - 44s - loss: 2.2465 - mean_absolute_error: 2.2464 - val_loss: 6.2013 -
val_mean_absolute_error: 6.2203
Epoch 15/16
178/178 - 41s - loss: 2.1821 - mean_absolute_error: 2.1820 - val_loss: 6.5504 -
val_mean_absolute_error: 6.5673
Epoch 16/16
178/178 - 40s - loss: 2.0547 - mean_absolute_error: 2.0546 - val_loss: 6.1170 -
val_mean_absolute_error: 6.1247
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
60/60 - 10s - loss: 6.1170 - mean_absolute_error: 6.1247
Test MAE: 6.1247
```

Анализ обученной модели

- Была создана нейронная сеть с архитектурой ResNet, с применением сверточных сетей, предобученых на ImageNet
- Обучение модели проходило в 16 эпох с алгоритмом Adam, где learning_rate = 0.0001

Итоговые результаты обучения:

- MAE на обучающей выборке: 2.0546
- MAE на валидационной выборке: 6.1247
- MAE на тестовой выборке: 6.1247

- Построенная модель по фото определяет возраст с ошибкой примерно 6 лет

Если использовать полученную модель, то ее достаточно для решения первой задачи (анализировать покупки и предлагать товары, которые могут заинтересовать покупателей определенной возрастной группы). При этом покупателей придется разбивать на группы с шагом не меньше 6 лет.

В случае второй задачи (принятие решение о продаже алкоголя) качества модели недостаточно для возрастной группы от 12 до 24 лет. В остальных возрастных группах использование модели приемлемо