**PetFinder.my Pawpularity Contest:**

**Предсказание популярности фотографии домашнего животного**

Дипломная работа

|  |  |
| --- | --- |
|  | Выполнил  Васильев Дмитрий Евгеньевич  Ментор  Миленкин Александр |

Красноярск - 2022

**СОДЕРЖАНИЕ**

Введение…………………………………………………………………………...4

1 Общее описание соревнования «Pawpularity Contest»………………………..5

1.1 Постановка задачи…………………………………………………………….6

1.2 Описание данных для обучения……………………………………….……..6

1.3 Метрики оценки качества…………………………………………………….6

2 Анализ данных…………………………………………………………………..6

2.1 Анализ распределения оценки привлекательности…………………………6

2.2 Анализ изображений………………………………………………………….9

3. Предобработка и поиск новых признаков…………………………………...11

3.1 Удаление дублированных изображений…………………………………...11

3.2 Генерация новых признаков с помощью детектора Yolov5………………14

3.2.1 Краткое описание механизма генерации новых признаков с помощью детекции………………………………………………………………………….14

3.2.2 Описание класса PetDataMining для генерации признаков и удаления дублей…………………………………………………………………………….14

3.2.3 Анализ сгенерированных признаков с помощью детекции…………….15

4 Описание модели………………………………………………………………16

4.1 Формирование датасета для обучения и теста…………………………….17

4.2 Экстрактор признаков изображений и полносвязная сеть………………..19

4.3 Способы обучение экстрактора признаков с полносвязной сетью, функции ошибок модели……………………………………………………..…20

4.3.1 Регрессионная модель без использования дополнительных метаданных……………………………………………………………………….20

4.3.2 Регрессионная модель c использования дополнительных метаданных..21

4.3.3 Классификационная модель без использования дополнительных метаданных……………………………………………………………………….22

4.3.4 Классификационная модель с использованием дополнительных метаданных……………………………………………………………………….22

4.4 Аугментация MixUP…………………………………………………………23

4.5 Модель градиентного бустинга……………………………………………..24

4.6 Поиск оптимальных гиперпараметров для модели………………………..25

5 Результаты……………………………………………………………………...26

5.1 Результаты обучения разных типов моделей……………………………...26

5.1.1 Результат регрессионной модели PetFinderTransferModel……………...27

5.1.2 Результат регрессионной модели PetFinderTransferModelWithAddFeatures……………………………………….29

5.1.4 Результат классификационной модели PetFinderTransferModelBCEWithAddFeatures………………………………….30

5.1.7 Результат наилучшей модели с блоком CatBoost……………………….32

5.2 Сводные результаты…………………………………………………………33

5.3 Пути развития………………………………………………………………..33

Список использованных источников…………………………………………..34

**Введение**

Оценка привлекательности по фотографии – одно из интересных направлений в области компьютерного зрения. Например, нейросеть Photofeeler D3 [1] умеет определять три качества человека по его фотографии: привлекательность, уровень интеллекта и безопасность. Данную модель используют сайты знакомств для рекомендаций улучшений фотографий для своих клиентов. Например, человек может запустить программу с данной нейронной сетью, а она в режиме реального времени выполняет оценку того, как человек выглядит в кадре с точки зрения привлекательности для противоположного пола. И таким образом человек сможет подобрать позу, одежду и аксессуары для получения максимального результата на сайте знакомств. Такие нейронные сети выглядят полезными и очень выгодными с точки зрения коммерции.

Так же данное направление может использоваться не только для коммерческих целей. Например, для помощи найти бездомному животному хорошего хозяина. Миллионы бездомных животных страдают на улицах или подвергаются эвтаназии в приютах каждый день по всему миру. Можно ожидать, что домашние животные с привлекательными фотографиями вызовут больший интерес и найдут своих будущих хозяев быстрее.

PetFinder.my - это ведущая платформа защиты животных Малайзии, в которой представлено более 180 000 животных. Компания предложила соревнование «Pawpularity Contest» на платформе Kaggle [2] с целью получения модели предсказания привлекательности домашних животных по фотографии. Целью данной работы является создание модели для данного соревнования.

**1 Общее описание соревнования «Pawpularity Contest»**

* 1. **Постановка задачи**

На соревновании [2] представлен датасет из 9912 фотографий с домашними животными. Для каждой фотографии существует оценка привлекательности / популярности (pawpularity) от 0 до 100.

Оценка популярности рассчитывалась на основе статистики просмотра страниц каждого профиля питомца на страницах списка с использованием алгоритма, который нормализует данные трафика с разных платформ (веб, мобильные). Повторяющиеся просмотры, посещения поисковых роботов и спонсируемые профили исключаются для анализа.

Требуется реализовать модель, которая сможет предсказать оценку популярности фотографии из данной методики расчета.

**1.2 Описание данных для обучения**

Организация предоставила метаданные визуального качества и композиции для каждой фотографии, которые были получены с помощью ручной разметки. При этом эти данные не участвуют в расчете оценки популярности.

Для каждой фотографии доступны следующие метаданные (принимают значение 0 или 1):

* **Subject Focus** (питомец хорошо различим на фотографии, не сливается с фоном);
* **Eyes** (оба глаза смотрят вперед или почти вперед);
* **Face** (опрятная мордочка, смотрит вперед или почти вперед);
* **Near** (одно домашнее животное, которое занимает более 50% ширины или высоты фотографии);
* **Action** (питомец выполняет действие, например, прыжок);
* **Accessory** (наличие на фото аксессуаров);
* **Group** (более 1 питомца на фото);
* **Collage** (фотография ретушью, обработкой и фотоэффектами);
* **Human** (на фото есть человек);
* **Occlusion** (наличие нежелательных предметов, блокирующие часть домашнего животного (клетка, забор и др.))
* **Info** (на фотографии присутствует текст);
* **Blur** (фотография размыта или зашумлена).

Метаданные и их взаимосвязь с фотографиями представлены в файле формата csv.

**1.3 Метрики оценки качества**

В качестве метрики оценки качества на соревновании используется формула RMSE:

где yi – исходное значение оценки привлекательности,

yi\_pred – предсказанное значение оценки привлекательности

**2 Анализ данных**

Все приведенные данные из раздела можно посмотреть в виде кода в ноутбуке [3] .

**2.1 Анализ распределения оценки привлекательности**

В первую очередь проанализируем оценку популярности.

Таблица 1 – Основные статистические метрики оценки популярности

|  |  |
| --- | --- |
|  | **Pawpularity** |
| **count** | 9912.000000 |
| **mean** | 38.039044 |
| **std** | 20.591990 |
| **min** | 1.000000 |
| **25%** | 25.000000 |
| **50%** | 33.000000 |
| **75%** | 46.000000 |
| **max** | 100.000000 |

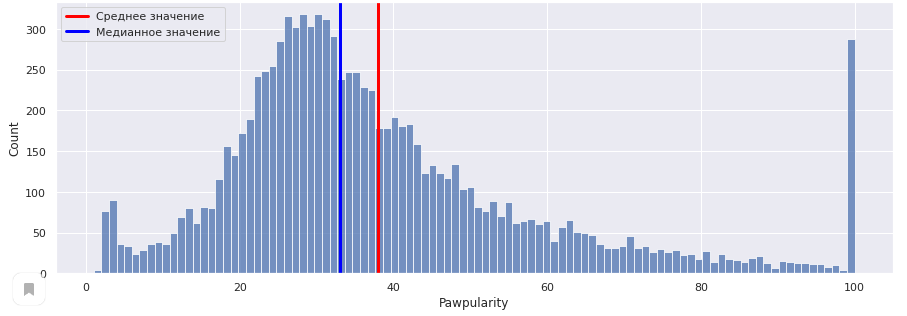


Рисунок 1 – Распределение оценки популярности (Pawpularity)

Из гистограммы распределений оценки популярности и основным статистических метрик можно сделать следующие выводы:

* Более 75% фотографий в наборе не привлекательны для потенциальных хозяев питомцев;
* Более 250 фотографий имеют аномальную оценку привлекательности 100.

В ходе экспериментов было выяснено, что удаление выбросов с оценкой 100 приводит к ухудшению результатов модели. Поэтому принято решение выбросы оставить. Результаты экспериментов приведены в разделе 5.2 данной работы.

**2.2 Анализ распределения метаданных**

Рассмотрим распределение метаданных относительно оценки привлекательности.

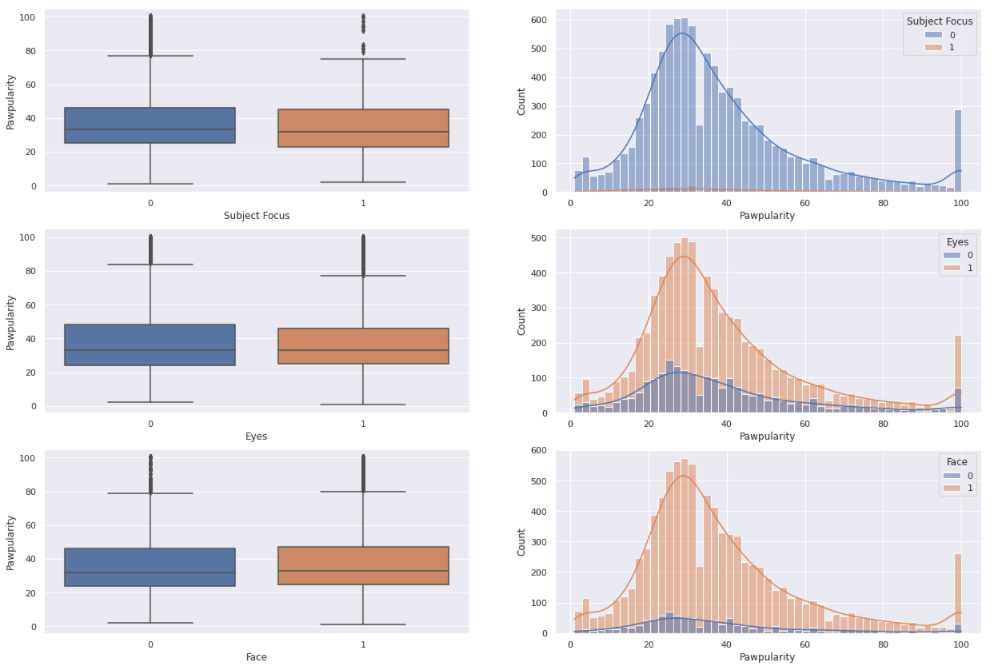


Рисунок 2 – Распределение метаданных относительно оценки популярности

На рисунке 2 приведены распределения для Subject Focus, Eyes и Face. Графики для всех метаданных приведены в [3].

Из полученных распределений видно, что метаданные практически не оказывают влияния на оценку популярности. В рамках каждого типа метаданных распределения разных значений практически совпадают. При этом наблюдается дисбаланс, значения 1 в рамках типов значительно превышают значения 0. Если строить модель только по метаданным, то хороших результатов добиться не получится. Поэтому основой для обучения модели будут непосредственно изображения питомцев.

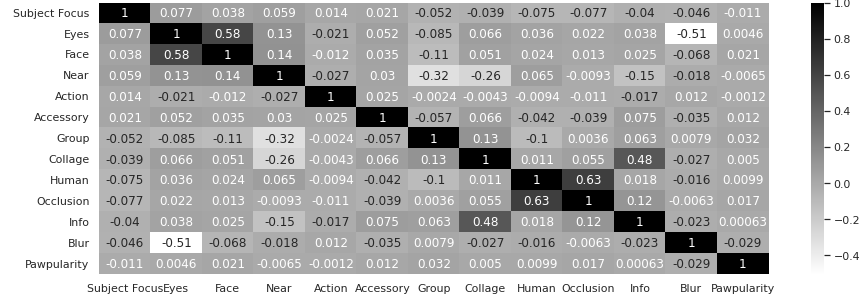


Рисунок 3 – Корреляция метаданных методом Спирмана

Аналогичный вывод можно сделать по таблице корреляции, ни один из типов метаданных не имеет корреляции с оценкой привлекательности.

**2.2 Анализ изображений**

Рассмотрим изображения в разрезе значений оценки привлекательности 2, 25, 75 и 100.

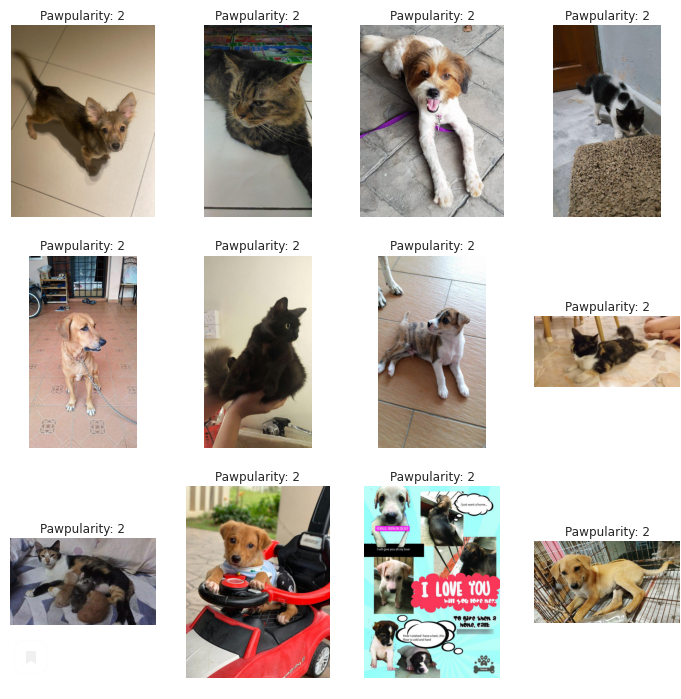


Рисунок 4 – Изображения с оценкой популярности 2

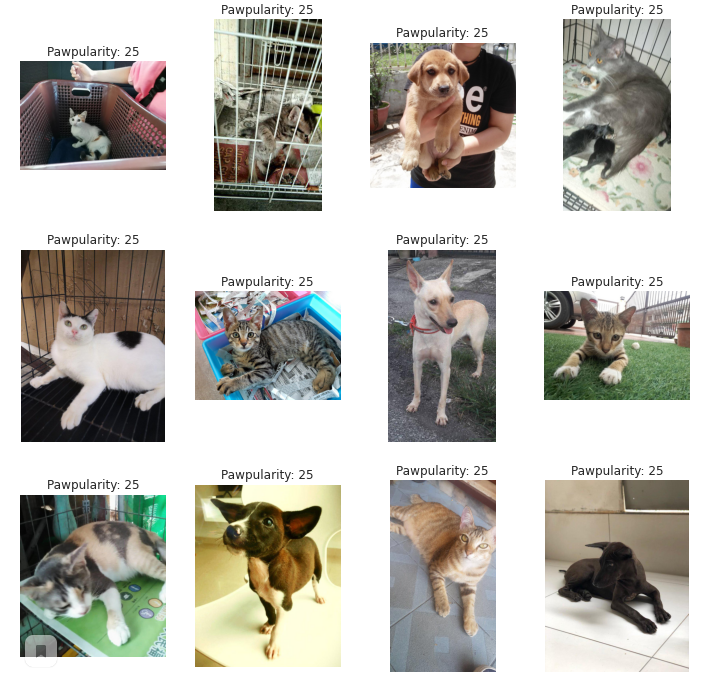


Рисунок 5 – Изображения с оценкой популярности 25

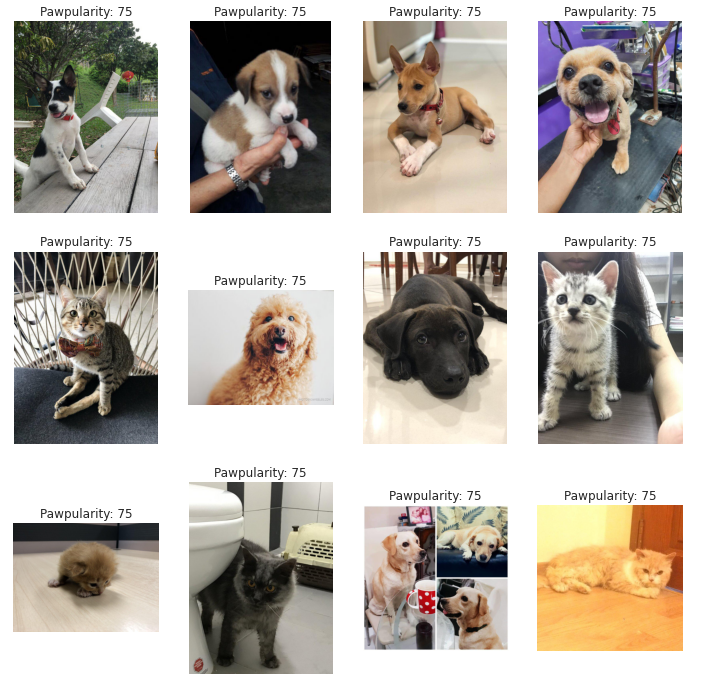


Рисунок 6 – Изображения с оценкой популярности 75

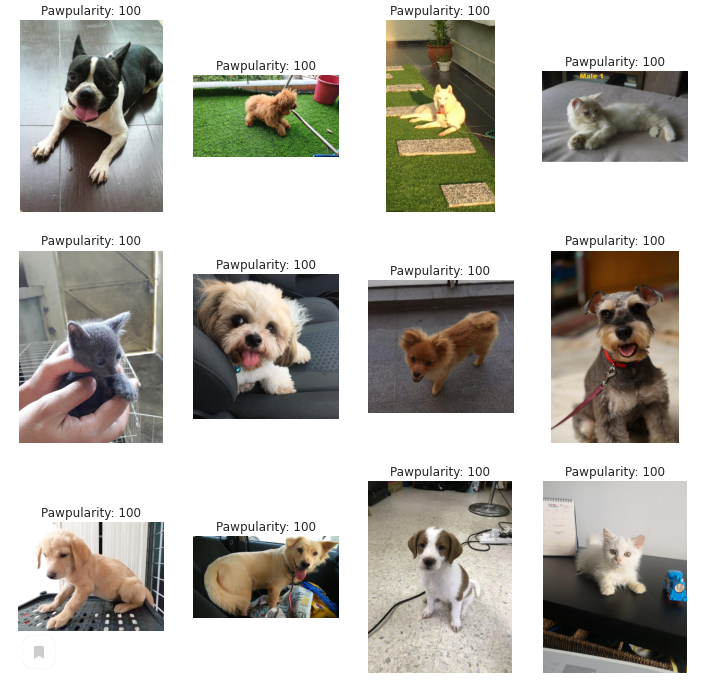


Рисунок 7 – Изображения с оценкой популярности 100

Из полученных изображений сложно дать категоричный вывод почему одни фотографии популярные, а другие нет. Но можно заметить тенденцию: фотографии более привлекательны, если питомец улыбается, имеет опрятный вид, имеет необычный аксессуар в виде бабочки на шее. Так же высокие оценки могут получать изображения с комичными ситуациями (строгий кот возле унитаза, забавное оптическое искажение мордочки собаки).

Так же сложно назвать причины, почему у более 250 фото (выброс) стоит оценка 100, фотографии в целом интересные и позитивные, но фото с оценками немного ниже в целом похожи. Поэтому данные фотографии можно попробовать удалить из обучающей выборки и сравнить результаты с моделью без удаления.

**3. Предобработка и поиск новых признаков**

**3.1 Удаление дублированных изображений**

В результате анализа датасета было установлено наличие дублей изображений. При этом дублированные изображения могут иметь разную оценку популярности. Принято решение об их удалении.

Для поиска одинаковых изображений можно с помощью сравнения перцептивных хэшей изображений (далее - phash). Алгоритмы хэширования изображений отличаются от криптографических тем, что при небольшом изменении картинки его хэш не поменяется. А криптографический хэш будет совершенно другим при малейшем изменении объекта. Таким образом алгоритмы хэширования изображений робастны даже к гамма-корреции и изменении цветовой гистраграммы.

Главная идея в хэшировании изображений заключается в выделении низкочастотных сигналов, на которых строится основной контур всех объектов на фотографии. Например, в перцептивном хэше используется алгоритм дискретного косинусного преобразования DCT [4].

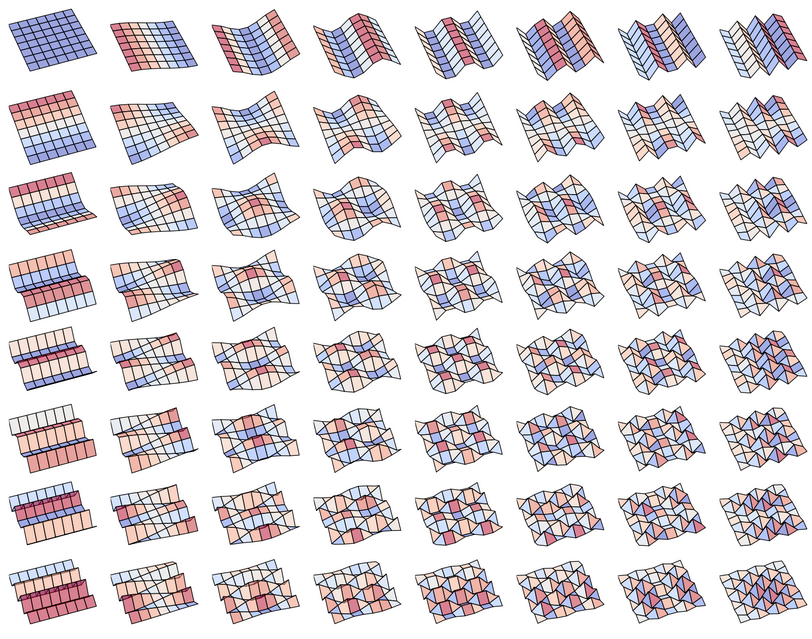


Рисунок 8 – Стандартные сигналы с разными частотами для DCT

Для преобразования DCT используется таблица стандартных косинусоидальных сигналов с разными частотами (рисунок 8). В левом верхнем углу находится сигнал с нулевой частотой. При движении в правую сторону и вниз частота сигнала увеличивается, даныне сигналы имеют разное направление. А по диагонали происходит сложение сигналов с разными частотами.

Алгоритм получения phash следующий [5]:

* сжать изображение до размера 32x32 (благодаря этой операции происходит первичное удаление высокочастотных составляющих изображения);
* убрать цвет фотографии для упрощения вычислений;
* применить дискретное косинусное преобразование и получить матрицу весов для каждого стандартного сигнала DCT;
* из матрицы весов DCT сохранить только левый верхний блок 8x8 (в нем хранятся веса для низкочастотных сигналов);
* вычислить среднее значение матрицы весов;
* преобразовать матрицу весов в значения из 1 и 0 (если значение больше среднего, то 1, иначе 0)
* на основе полученного преобразования построить хэш в виде 64-битного числа.

Два хэша изображений можно сравнить между собой с помощью расстояния Хэмминга:

Ниже на рисунке представлен пример найденных дублей. Реализацию можно посмотреть в [6].

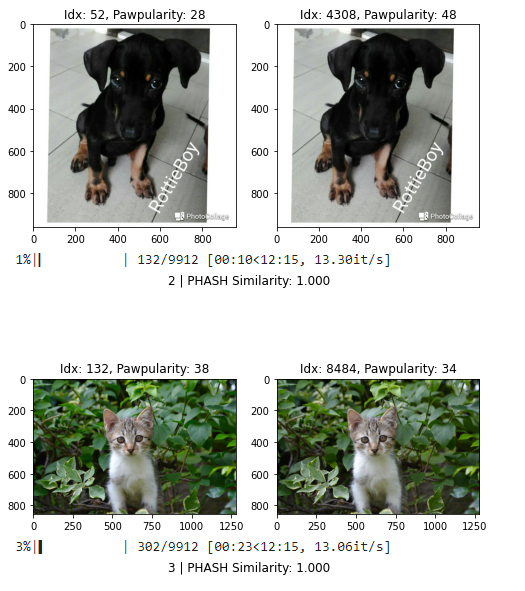


Рисунок 9 – Пример найденных дублей с помощью сравнения phash с порогом 0.9

**3.2 Генерация новых признаков с помощью детектора Yolov5**

**3.2.1 Краткое описание механизма генерации новых признаков с помощью детекции**

Для обучения модели с помощью детектора предлагается сгенерировать следующие признаки:

* n\_pets (количество кошек и собак на изображении);
* is\_unknown (равен 1, если на изображении не найдены собаки и кошки, иначе 0);
* is\_cat (равен 1, если самая достоверная детекция на фото - кошка, иначе 0);
* is\_dog (равен 1, если самая достоверная детекция на фото - собака, иначе 0);
* x\_min - минимальное значение по оси x в найденных боксах;
* x\_max - максимальное значение по оси x в найденных боксах;
* y\_min - минимальное значение по оси y в найденных боксах;
* y\_max - максимальное значение по оси y в найденных боксах;
* pet\_ratio - соотношение размера животных на фото к размеру изображения.

Так как практически на всех фото находятся только кошки или собаки, то будем формировать признаки именно по ним. В случае если на фото не найдена ни кошка, ни собака, будет формироваться признак is\_unknown.

Признак pet\_ratio можно посчитать как соотношение площади всех bounding box кошек и собак, полученных с помощью детектора, к площади размера изображения.

Данные признаки можно сгенерировать как для обучающего датасета, так и для тестового в закрытой части соревнования.

**3.2.2 Описание класса PetDataMining для генерации признаков и удаления дублей**

Для удаления дублей изображений и генерации новых признаков с помощью детектора Yolov5 создан класс PetDataMining.

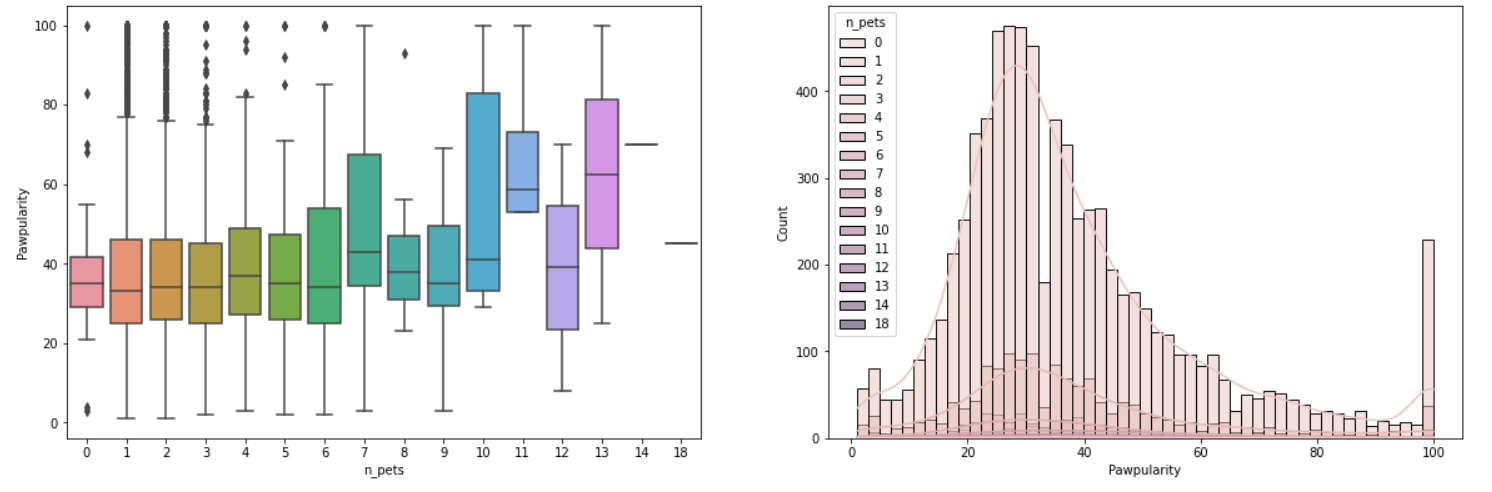
Метод get\_images\_hash() добавляет в датасет phash. А метод find\_similar\_images() позволяет найти идентификаторы дублей изображений. Метод detector\_data\_mining() запускает генерацию признаков для всех изображений. С помощью метода start() можно запустить операцию удаления дублей и генерацию признаков за раз.

После инициализации класса в Jupyter Notebook необходимо выполнить команду %matplotlib inline . Это связано с тем, что загрузка детектора с помощью TorchHub меняет настройки matplotlib и приводит   
к неожиданному поведению команды plt.show() .

Документированный код класса можно найти в [7].

**3.2.3 Анализ сгенерированных признаков с помощью детекции**

Проанализируем новые признаки n\_pets (количество животных на фото) и pet\_label (метка unknown, cat или dog).

****

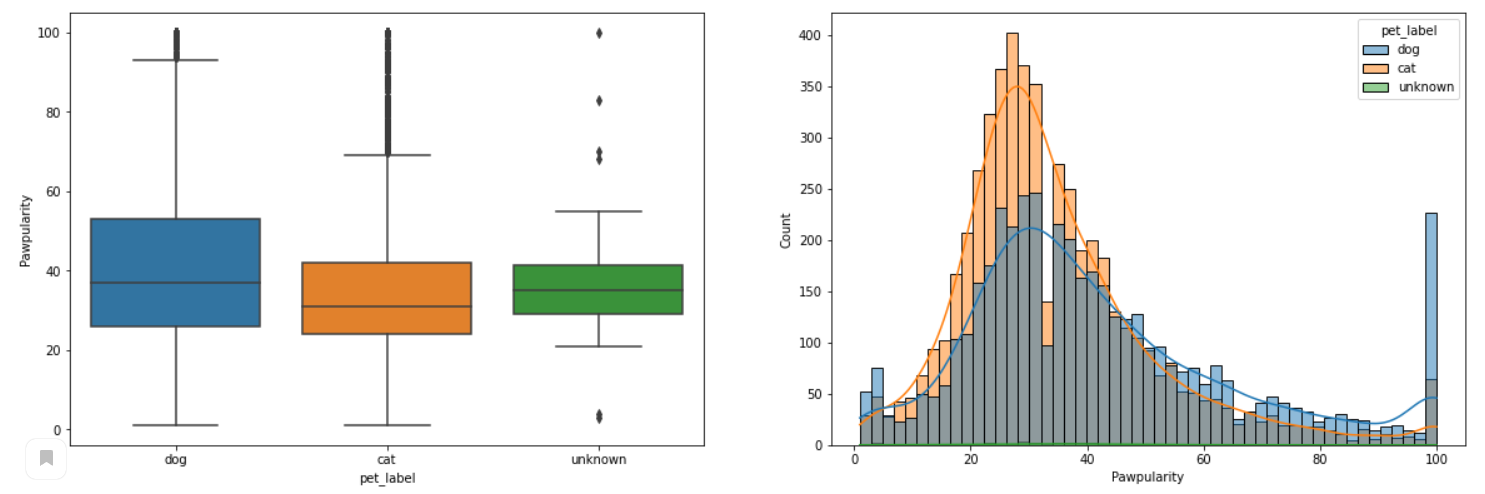
****

Рисунок 10 – Статистические данные по n\_pets и pet\_label

Из графиков можно сделать вывод, что собаки получают оценки лучше, чем кошки. И примерно одинаковое количество оценок от 0 до 20. Так же более высокие оценки получают фото с 11 домашними животными. Но здесь может быть погрешность из-за малого объема данных, так как фото с 12 животными получает в целом средние оценки. При этом вряд ли людей могут привлекать фото именно с 11 животными, а не с 12. Вероятно в этих фотографиях есть другие отличительные признаки.

Исходя из графиков можно предположить, что дополнительные данные могут улучшить обучение модели, так как распределения данных в признаках различны. Код с анализом дополнительных признаков представлен в [6].

**4 Описание модели**

На слайде приведена схема результирующей ансамблевой модели. Ансамбль состоит из блока основной модели и дополнительной. Дополнительная модель является опциональной и была введена для улучшения результатов основной модели.

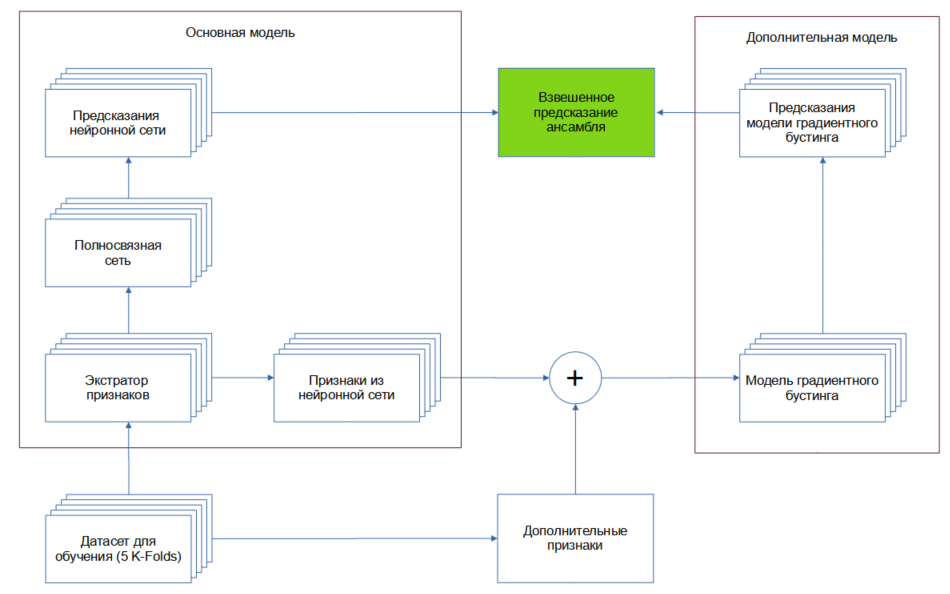
****

Рисунок 11 – Общая схема ансамблевой модели

Основная модель состоит из экстрактора признаков и полносвязной сети на его выходе. В качестве экстрактора признаков может выступать любая архитектура. Тип архитектуры экстрактора признаков является гиперпараметром, который можно подбирать и менять на лету без ущерба другим частям модели.

Дополнительная модель состоит из блока моделей градиентного бустинга. Как работает ансамблевая модель.

Вначале данные разбиваются на пять фолдов, для каждого фолда обучается отдельная модель экстрактора признаков и полносвязной сети. Если дополнительная модель не подключена, то с выхода полносвязной сети с каждого фолда получаем предсказания и усредняем его по количеству фолдов.

Если подключена дополнительная модель, то на ее вход поступают выходы каждого фолда экстрактора признаков. При этом они конкатенируются с дополнительными признаками из датасета. В результате оценка будет усредняться вначале по фолдам в рамках каждой модели. А потом будет происходить усреднение предсказания каждой модели с помощью весов.

Модель будет строиться на базе фреймворков PyTorch и PyTorch Lighting. В следующих подразделах рассмотрим устройство ансамбля более подробно.

**4.1 Формирование датасета для обучения и теста**

На данном этапе формируются тензоры PyTorch для использования на входе экстрактора признаков. Предварительно в датасете обучения удаляются все дубликаты с помощью класса PetDataMining с порогом 0.9 и генерируются признаки с помощью детектора Yolov5. Для тестового датасета с закрытыми данными применяется только генерация признаков.

Обучающий датасет будет разбиваться на 5 фолдов с помощью класса StratifiedKFold из фреймворка sklearn. Каждый фолд содержит часть для обучения и часть для валидации. На каждом фолде обучается отдельная модель экстрактора признаков и модель полносвязной сети.

За формирование данных для обучения модели отвечают следующие классы:

* PetDataSet (наследуется от класса Dataset фреймворка PyTorch);
* PetDataLoader (наследуется от класса LightningModule фреймворка PyTorch Lighting)

PetDataSet отвечает за формирование и хранение данных для обучения. А так же за их предварительную обработку. PetDataLoader отвечает за загрузку данных в модель, формирование батчей и разбиение данных на фолды.

Класс PetDataSet выполняет следующие функции:

* аугментация изображений;
* стандартизация дополнительных признаков;
* применение к дополнительным признакам преобразования PCA;
* разделение датасета на части для обучения, валидации и предсказания;
* формирование и хранение данных для использования в модели.

В качестве аугментации используется случайный поворот по вертикали и горизонтали, случайное удаление фрагментов из изображения. Также в одной из вариации модели используется аугментация MixUp. Так как данный вид аугментации использует свою уникальную функцию ошибок, она была реализована в одном из классов модели и может включаться опционально. Об этом будет рассказано в разделе описания модели.

Стандартизация и преобразование PCA являются опциональными. Если включить оба преобразования, то вначале будет выполнена стандартизация, а после уже преобразование PCA. При этом PCA не изменит количество дополнительных признаков. Такая операция позволит получить ортогональные вектора.

Класс PetDataSet является итерируемым объектом и инициализируется датафреймом с данными для обучения и тестирования. С помощью метода split() класс меняет свое состояние и во время итерации выдает либо данные для обучения, валидации или для теста. Данные для теста – данные для предсказания из закрытой публичной части. Для инициализации класса необходимо предварительно в датафрейме части обучения разметить каждую строку с признаками фото на типы «train» или «val» с помощью колонки «split». А данные из зарытой публичной части добавить в обучающий датафрейм с меткой «predict». Документированный код класса можно найти  
 в [7].

Класс PetDataLoader выполняет следующие функции:

* инициализация и использование класса PetDataSet;
* разбиение данных на фолды с помощью StratifiedKFold фреймворка Scikit-learn;
* расчет весов для обучающих данных;
* создание загрузчиков данных для обучения, валидации и теста;
* генерация загрузчиков для Test Time Augmentation (TTA).

Класс PetDataLoader наследуется от LightningDataModule фрейморка PyTorch Lighting. С помощью метода create\_kfold\_loaders() можно инициализировать необходимое количество экземпляров PetDataLoader для каждого стратифицированного фолда. Метод самостоятельно разобьет данные и вернет список из экземпляров PetDataLoader. Количество объектов PetDataLoader равно количеству указанных фолдов в параметре n\_splits указанного метода. Также метод create\_kfold\_loaders() вычисляет веса для каждого экземпляра датасета. Документированный код класса можно найти   
в [6].

Структура батча данных датасета:

{

'hash': хэш изображения,

'x\_data': Тензор изображения размером N x C x H x W,

'additional\_features': дополнительные метаданные,

'image\_path': путь к файлу изображения,

'pawpularity': оценка привлекательности изображения,

'class\_weight': вес класса изображения

}

**4.2 Экстрактор признаков изображений и полносвязная сеть**

В качестве экстрактора признаков можно использовать самые различные архитектуры нейронных сетей. В реализованном коде данной работы модель нейронной сети для экстрактора признаков является гиперпараметром. С помощью конфигурации можно менять модель экстрактора признаков и проверять итоговое качество общей модели.

Для загрузки разных типов нейронных сетей для экстратора признаков реализован вспомогательный класс TransferNet.

Данный класс может создать экстрактор признаков на основе следующих моделей:

* ResNet18;
* VGG16;
* GoogleNet;
* EfficientNetV2;
* SWIN Transformer;
* ViT.

В качестве параметра для инициализации модели можно указать структуру полносвязной сети на выходе экстрактора признака или задать ее отсутствие.

Например, с помощью следующей инициализации мы получим SWIN Transformer, у которого на выходе полносвязная сеть с тремя слоями из 128, 256 и 1 элемента соответственно:

model\_gen = TransferNet(output\_dims=[128, 256, 1], dropout=0.5, pretrained=True, full\_trainable=True, seed=42)

model = model\_gen. make\_model('swin\_base\_patch4\_window7\_224')

Отсюда следует, что структура полносвязной сети на выходе экстрактора признаков тоже является гиперпараметром и ее можно оптимизировать. Класс TransferNet создан именно для генерации модели, но не для ее оптимизации. Документированный код класса можно посмотреть в [7].

**4.3 Способы обучение экстрактора признаков с полносвязной сетью, функции ошибок модели**

В зависимости от того, каким способом будет решаться задача, можно формировать различную структуру полносвязной сети для экстрактора признаков. Отличаться будут и функции ошибок для оптимизации.

**4.3.1 Регрессионная модель без использования дополнительных метаданных**

В данном случае последний слой полносвязной сети должен состоять из одного элемента. А в качестве функции ошибок будет использоваться RMSE:

Для реализации данной связки экстрактора признаков и полносвязной сети реализован класс PetFinderTransferModel, который наследуется от класса LightningModule фреймворка PyTorch Lighting.

Пример инициализации модели:

model = PetFinderTransferModel (model\_name='swin\_base\_patch4\_window7\_224', output\_dims=[1024, 512, 256, 1], dropout=0.5, learning\_rate=0.0001, full\_trainable=True, l2\_regularization=1e-3, adam\_betas=(0.9, 0.999), pretrained=True, seed=2022)

Модель PetFinderTransferModel, как и и все последующие реализации других моделей, используют класс TransferNet для генерации экстрактора признаков и полносвязной сети. В данном модели важно указать в параметре output\_dims последнее значение в виде 1, иначе произойдет исключение. К последнему слою полносвязной сети не применяется функция активации.

Наследуемые от LightningModule классы используются для создания обучающейся модели. Поэтому в PetFinderTransferModel и в других подобных классах реализованы шаги обучения, определена функция оптимизации, функция планировщика обучения, функция печати графика обучения в конце каждой эпохи. А так же реализованы функции ошибок.

В качестве метода оптимизации для всех моделей выбран AdamW, как один из самых быстрых и эффективных методов. При инициализации класса модели можно указать параметры коэффициентов оптимизатора AdamW.

Документированный код класса можно посмотреть в [7].

**4.3.2 Регрессионная модель c использования дополнительных метаданных**

Данная модель отличается от модели из 4.3.1 только тем, что к выходу экстрактора признаков присоединяется вектор дополнительных признаков (стандартные и сгенерированные с помощью детектора YoloV5).

Для реализации данной модели сделан класс PetFinderTransferModelWithAddFeatures, который наследуется от PetFinderTransferModel. Основное отличие – класс самостоятельно изменяет структуру сгенерированной модели. В частности, вносит изменения в размер входа полносвязной сети (сумма размера выхода экстрактора признаков и размера дополнительных метаданных). И во время обучения соединяет выход экстрактора признаков с тензором дополнительных метаданных и подает на вход полносвязной сети.

Документированный код класса можно посмотреть в [7].

**4.3.3 Классификационная модель без использования дополнительных метаданных**

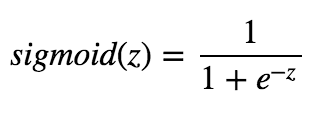
В данном случае последний слой полносвязной сети должен аналогично состоять из одного элемента. А в качестве функции ошибок будет использоваться Binary Cross Entropy:



где yi – целевое значение,

p(yi) – предсказанное значение

Для использования данной функции ошибок целевую переменную необходимо перевести в диапазон от 0 до 1. Для этого оценку популярности требуется разделить на 100. А к последнему слою нейронной сети применить сигмоидальную функцию активации:



Для реализации данной модели создан класс PetFinderTransferModelBCE, наследуемый от PetFinderTransferModel. Документированный код класса можно посмотреть в [7].

**4.3.4 Классификационная модель с использованием дополнительных метаданных**

Модель аналогична модели из 4.3.3, только в ней по аналогии из 4.3.2 к признакам экстрактора добавляются дополнительные метаданные.

Для реализации этой модели создан класс PetFinderTransferModelBCEWithAddFeatures, который наследуется от PetFinderTransferModelBCE и PetFinderTransferModelWithAddFeatures. Документированный код класса можно посмотреть в [7].

**4.4 Аугментация MixUP**

С помощью аугментации MixUp происходит смешивание двух изображений из разных классов в одно изображение. Данная техника описана в [8] .

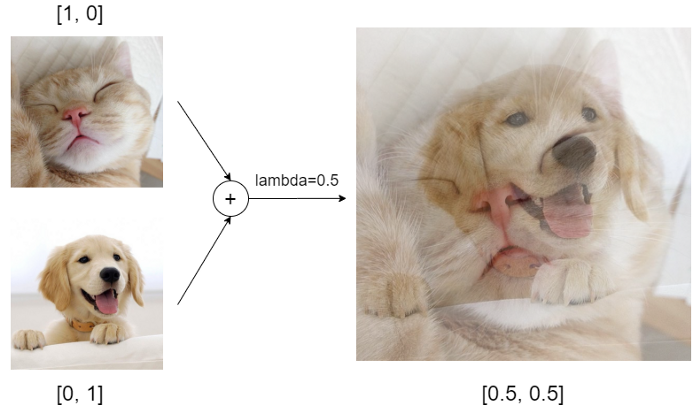


Рисунок 12 – Демонстрация техники MixUp

Смешивание изображений происходит по следующей формуле:

где xi – первое изображение,

xj – второе изображение,

– случайное число из бета-распределения.

Коэффициенты для бета-распределения берутся в диапазоне от   
 0 < < 1 . В результате график функции бета-распределения выглядит, как на рисунке

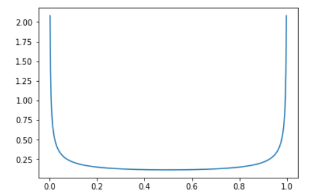


Рисунок 13 – График бета-распределения с коэффициентами от 0 до 1.

Так же изменяется функция ошибок для оптимизации модели:

где Li – функция ошибок первого изображения,

Lj – функция ошибок второго изображения,

– случайное число бета-распределения.

В функции ошибок используются исходные метки изображений. А на вход экстрактора признаков подается комбинированное изображение.

Техника MixUp реализована только в классах моделей PetFinderTransferModelBCE и PetFinderTransferModelWithAddFeaturesBCE.

При инициализации классов можно указать следующие параметры:

- use\_mixup (по умолчанию False);

- mixup\_alpha (по умолчанию 0.5);

- mixup\_prob (по умолчанию 0.5).

Параметр use\_mixup включает или выключает MixUp. А с помощью параметра mixup\_prob указывается с какой вероятностью будет использован MixUp или обычная функция ошибок без аугментации MixUp. Документированный код классов можно посмотреть в [7].

**4.5 Модель градиентного бустинга**

В качестве модели градиентного бустинга был выбран CatBoost со следующими гиперпараметрами:

{'loss\_function' : 'RMSE',

                     'eval\_metric' : 'RMSE',

                     'iterations' : 20000,

                     'od\_type': 'Iter',

                     'od\_wait': 200,

                     'learning\_rate' : 0.002}

Какие гиперпараметры CatBoost могут значительно влиять на качество [14]:

- sample\_rate (бустрап документов с методом Бернулли);

- bagging\_temperature (Байесовский бустрап документов, интенсивность семплирования);

- random\_strength (коэффициент для прибавления к score сплита случайного значения, которое уменьшается с увеличением количества итераций)**.**

С данными параметрами можно экспериментировать, в данной работе эти параметры взяты со значением по умолчанию.

Так же включен детектор переобучения с помощью параметров od\_type и od\_wait. Модель автоматически остановится, когда ошибка на валидации не будет уменьшаться в течении количества итераций из od\_wait.

**4.6 Поиск оптимальных гиперпараметров для модели**

Поиск оптимальных параметров осуществляется через фреймворк Optuna [9].

Преимущества использования Optuna:

- использует байесовские алгоритмы для оптимизации параметров;

- возможность ранней остановки неперспективного эксперимента исходя из статистики (например, если результаты эксперимента ниже медианы из уже рассмотренных);

- хорошая документация;

- возможность хранения результатов в базе данных, восстановление поиска гиперпараметров из базы данных;

- параллельный поиск параметров с разных устройств с помощью базы данных;

- один из самых быстрых фреймворков оптимизации гиперпараметров (в 1.5 раза быстрее hyperopt) [10] .

В ноутбуке [11] представлен пример оптимизации следующих гиперпараметров:

- количество слоев в полносвязной сети на выходе экстрактора признаков;

- количество нейронов в каждом слое полносвязной сети;

- значение dropout в полносвязной сети;

- скорость обучения learning\_rate;

- регуляризация l2\_regularization;

- значения beta1 и beta2 для AdamW оптимизатора;

- включение/отключение MixUp;

- включение/отключение аугментации;

- включение стандартизации дополнительных признаков std\_for\_add\_features;

- включение преобразования PCA для дополнительных признаков pca\_for\_add\_features.

Гиперпараметры подбираются для модели без блока CatBoost.

**5 Результаты**

**5.1 Результаты обучения разных типов моделей**

Данные раздел ставит цель сравнения между собой всех типов реализованных моделей.

Все модели будут обучаться со следующими гиперпараметрами:

model\_name: swin\_base\_patch4\_window7\_224\_in22k,

output\_dims: [1022, 1]

use\_weights: False

dropout: 0.3232967040113801

learning\_rate: 4.812096281914617e-06,

l2\_regularization: 0.2766337962203344

adam\_betas: (0.37438095497909607, 0.7504122450294624)

pca\_for\_add\_features: False

std\_for\_add\_features: True

p\_vflip: 0.5

p\_hflip: 0.5

Данные гиперпараметры были подобраны с помощью Optuna на модели PetFinderTransferModelBCEWithAddFeatures. Так как нам необходимо сравнить эффективность моделей между собой, то необходимо для всех моделей взять максимально одинаковые гиперпараметры. Были проверены стандартные гиперпараметры, они показали себя хуже, чем подобранные с помощью Optuna.

Если модель использует дополнительные признаки, то используются стандартные и сгенерированные вместе.

Так же все приведенные графики обучения для моделей будут приведены только для первого фолда. Данные в фолдах фиксируются в помощью фиксации генератора случайных чисел. Модель останавливает свое обучение, если в течении 3 эпох значение функции ошибки RMSE не уменьшается более чем на 0.01.

Для некоторых моделей может проводится несколько экспериментов. Если их конфигурация будет отличаться от указанной выше, об этом будет сделан комментарий. Все модели обучаются и проверяются без блока с моделями CatBoost, если не указано иное.

**5.1.1 Результат регрессионной модели PetFinderTransferModel**

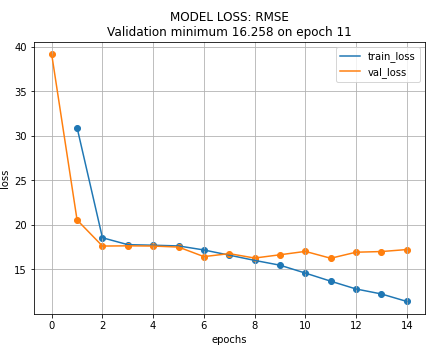
****

Рисунок 11 – График обучения модели PetFinderTransferModel с удалением выбросов

Для итоговой модели на первом фолде взяты веса из 11 эпохи с минимальным RMSE на валидации. Отсутствует MixUp, ТТА и блок моделей CatBoost. Были удалены выбросы с оценкой привлекательности 100.

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 14.114

Результаты модели на Kaggle:

Private Score: 18.11833

Public Score: 18.58314

Результаты аналогичной модели, но без удаления выбросов с оценкой популярности 100:

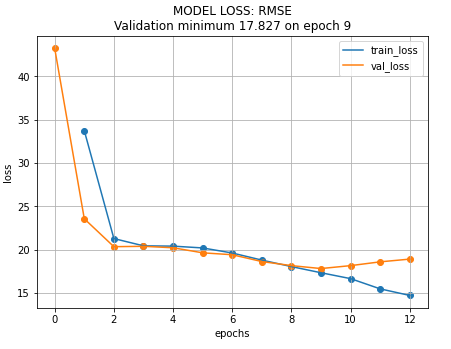


Рисунок 12 – График обучения модели PetFinderTransferModel без удаления выбросов

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 15.34

Без удаления выбросов результаты обучения на валидационном датасете немного ухудшились, но улучшились при получении результатов на Kaggle в закрытой части.

Результаты модели на Kaggle:

Private Score: 17.82002

Public Score: 18.28420

Отсюда можно сделать вывод, что данные с выбросом оценки в значении 100 удалять не нужно.

**5.1.2 Результат регрессионной модели PetFinderTransferModelWithAddFeatures**

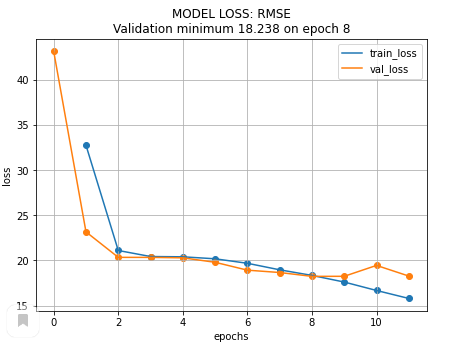
****

Рисунок 13 – График обучения модели PetFinderTransferModelWithAddFeatures

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 15.34

Результаты модели на Kaggle:

Private Score: 18.06978

Public Score: 18.42875

**5.1.3 Результат классификационной модели PetFinderTransferModelBCE**

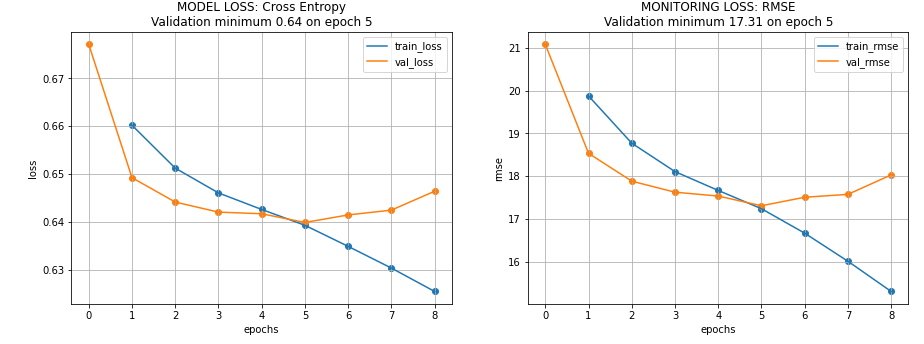
****

Рисунок 14 – График обучения модели PetFinderTransferModelBCE без использования MixUp

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 15.96

Результаты модели на Kaggle:

Private Score: 17.55870

Public Score: 18.24565

Теперь проведем эксперимент с включенным MixUp. Зададим параметры mixup\_alpha = 0.5, mixup\_prob = 0.5.

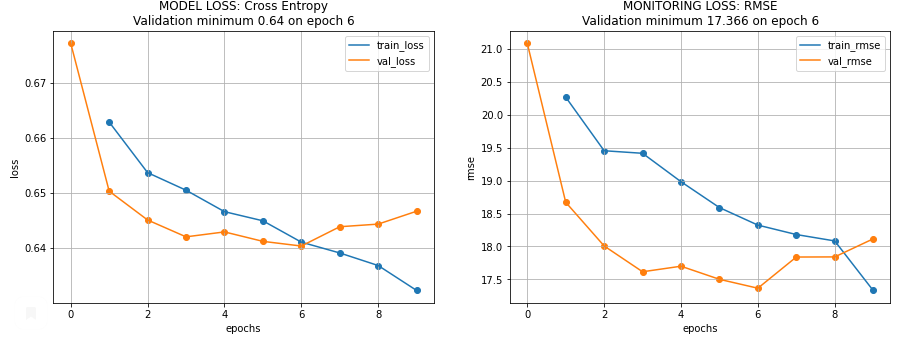


Рисунок 15 – График обучения модели PetFinderTransferModelBCE с использованием MixUp

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 15.96

Результаты модели на Kaggle:

Private Score: 17.48160

Public Score: 18.21914

С включенным MixUp результаты модели улучшились.

**5.1.4 Результат классификационной модели PetFinderTransferModelBCEWithAddFeatures**

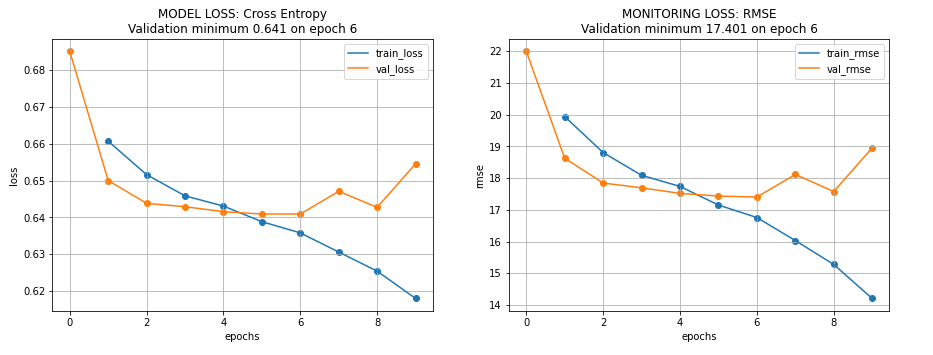
****

Рисунок 16 – График обучения модели PetFinderTransferModelBCEWithAddFeatures

без использования MixUp

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 15.34

Результаты модели на Kaggle:

Private Score: 18.06978

Public Score: 18.42875

Теперь проведем эксперимент с включенным MixUp. Зададим параметры mixup\_alpha = 0.5, mixup\_prob = 0.5.

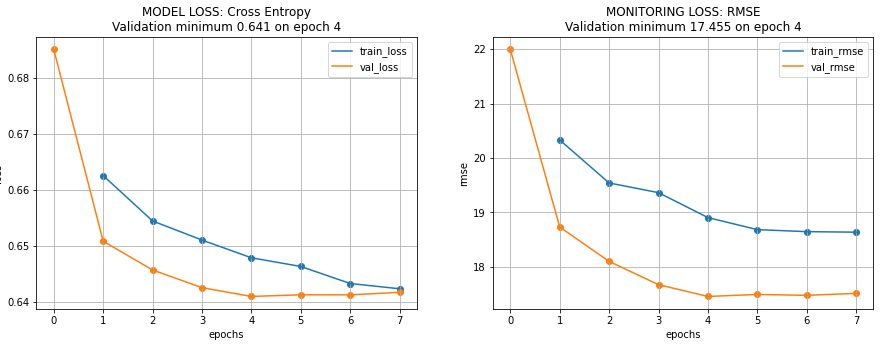


Рисунок 17 – График обучения модели PetFinderTransferModelBCEWithAddFeatures с использованием MixUp

Среднее значение RMSE для всех K-FOLD моделей на валидационном датасете: 16.22

Результаты модели на Kaggle:

Private Score: 17.49111

Public Score: 18.22110

С включенным MixUp результаты модели значительно улучшились.

**5.1.7 Результат наилучшей модели с блоком CatBoost**

Наилучший результат в Private Score получила модель PetFinderTransferModelBCE с включенным MixUp. Данная модель не использует дополнительные признаки. Дополнительные признаки будем передавать только в модель CatBoost. Вначале обучается модель экстрактор признаков с полносвязным слоем с помощью функции ошибок Binary Cross Entropy. После этого с выхода экстрактора признаков получаются признаки изображения для входа блока моделей Catboost. И к ним присоединяются дополнительные признаки.

Взяты следующие веса для предсказания моделей:

- 0.5 для нейронной сети;

- 0.5 для CatBoost.

Результаты модели на Kaggle:

Private Score: 17.46633

Public Score: 18.25116

По сравнению с моделью без CatBoost результаты улучшены с 17.48160 до 17.46633.

Код для обучения данной модели можно найти в ноутбуке [12]. А веса для всех моделей обученных моделей доступны по ссылке [13] .

**5.2 Сводные результаты**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Модель** | **Результат в закрытой приватной части** | **Результат в открытой приватной части** | **Комментарий** |
| 1 | Регрессионная модель PetFinderTransferModel (тип 1) | 18.11833 | 18.58314 | Удалены выбросы с оценкой 100 |
| **2** | Регрессионная модель PetFinderTransferModel (тип 2) | 17.82002 | 18.28420 | Без удаления выборосов |
| **3** | Регрессионная модель PetFinderTransferModelWithAddFeatures | 18.06978 | 18.42875 |  |
| 4 | Классификационная модель PetFinderTransferModelBCE (тип 1) | 17.55870 | 18.24565 | Выключен MixUp |
| 4 | Классификационная модель PetFinderTransferModelBCE (тип 2) | 17.48160 | 18.21914 | Включен MixUp |
| 5 | Классификационная модель PetFinderTransferModelBCEWithAddFeatures (тип 1) | 17.52333 | 18.32462 | Выключен MixUp |
| 6 | Классификационная модель PetFinderTransferModelBCEWithAddFeatures (тип 2) | 17.49111 | 18.22110 | Включен MixUp |
| 7 | Классификационная модель PetFinderTransferModelBCE (тип 2) + CatBoost | 17.46633 | 18.25116 | Включен MixUp для нейронной сети |

**5.2 Пути развития**

* Попробовать более мощные SWIN с большим разрешением изображения на входе
* Попробовать более сложные ансамбли с более разнообразными моделями
* Удалить из обучения стандартные признаки, оставить только признаки детектора
* Использовать для обучения библиотеку Fast.AI (хорошие встроенные функции агументации и TTA)

**Список использованных источников**

[1] Photofeeler-D3: A Neural Network with Voter Modeling for Dating Photo Impression Prediction (https://arxiv.org/abs/1904.07435)

[2] PetFinder.my – Pawpularity Contest (https://www.kaggle.com/c/petfinder-pawpularity-score/overview)

[3] Ноутбук с анализом данных для PetFinder.me (https://github.com/inject0r1945/ds\_projects/blob/main/kaggle/PetFinder.me\_Pawpularity\_Contest/petfinder\_EDA.ipynb)

[4] JPEG DCT, Discrete Cosine Transform (https://www.youtube.com/watch?v=Q2aEzeMDHMA)

[5] «Выглядит похоже». Как работает перцептивный хэш (https://habr.com/ru/post/120562/)

[6] Ноутбук с демонстрацией генерации новых признаков и удаления дубликатов (https://github.com/inject0r1945/ds\_projects/blob/main/kaggle/PetFinder.me\_Pawpularity\_Contest/petfinder\_data\_mining.ipynb)

[7] Разработанная библиотека для построения модели (https://github.com/inject0r1945/ds\_projects/blob/main/kaggle/PetFinder.me\_Pawpularity\_Contest/pet\_finder\_library.py)

[8] mixup: Beyond Empirical Risk Minimization (https://arxiv.org/abs/1710.09412v2)

[9] Optuna (https://optuna.readthedocs.io/en/stable/tutorial/index.html)

[10] Гиперпараметры: как перестать беспокоиться и начать их оптимизировать (https://habr.com/ru/company/antiplagiat/blog/528384/)

[11] Пример оптимизации модели с Optuna (<https://github.com/inject0r1945/ds_projects/blob/main/kaggle/PetFinder.me_Pawpularity_Contest/petfinder_data_mining.ipynb>)

[12] Ноутбук с демонстрацией финальной модели (https://github.com/inject0r1945/ds\_projects/blob/main/kaggle/PetFinder.me\_Pawpularity\_Contest/petfinder\_final\_model.ipynb)

[13] Файлы весов для обученных в ноутбуке моделей (https://disk.yandex.ru/d/IZotUcQa\_1aw6Q)

[14] CatBoost - градиентный бустинг от Яндекса (https://www.youtube.com/watch?v=UYDwhuyWYSo)