PETFINDER PAWPULARITY

PREDICTING THE APPEAL OF PET IMAGES

ABHISHIKTH PERI
AMULYA KONDA
KASTURI PAL
SANDEEP VENKATA
SHARMISHTA PATRA

CARLSON SCHOOL OF MANAGEMENT
UNIVERSITY OF MINNESOTA

Agenda

- Situation
- Complication
- Key Question
- Data structure
- Strategy
- Modeling
- Performance
- Insights

Situation

PetFinder.my, Malaysia’s leading animal welfare platform, currently employs a basic Cuteness Meter to rank pet photos. This meter analyzes picture composition and other factors to evaluate the appeal of pet profiles, with over 180,000 animals featured and 54,000 successfully adopted.

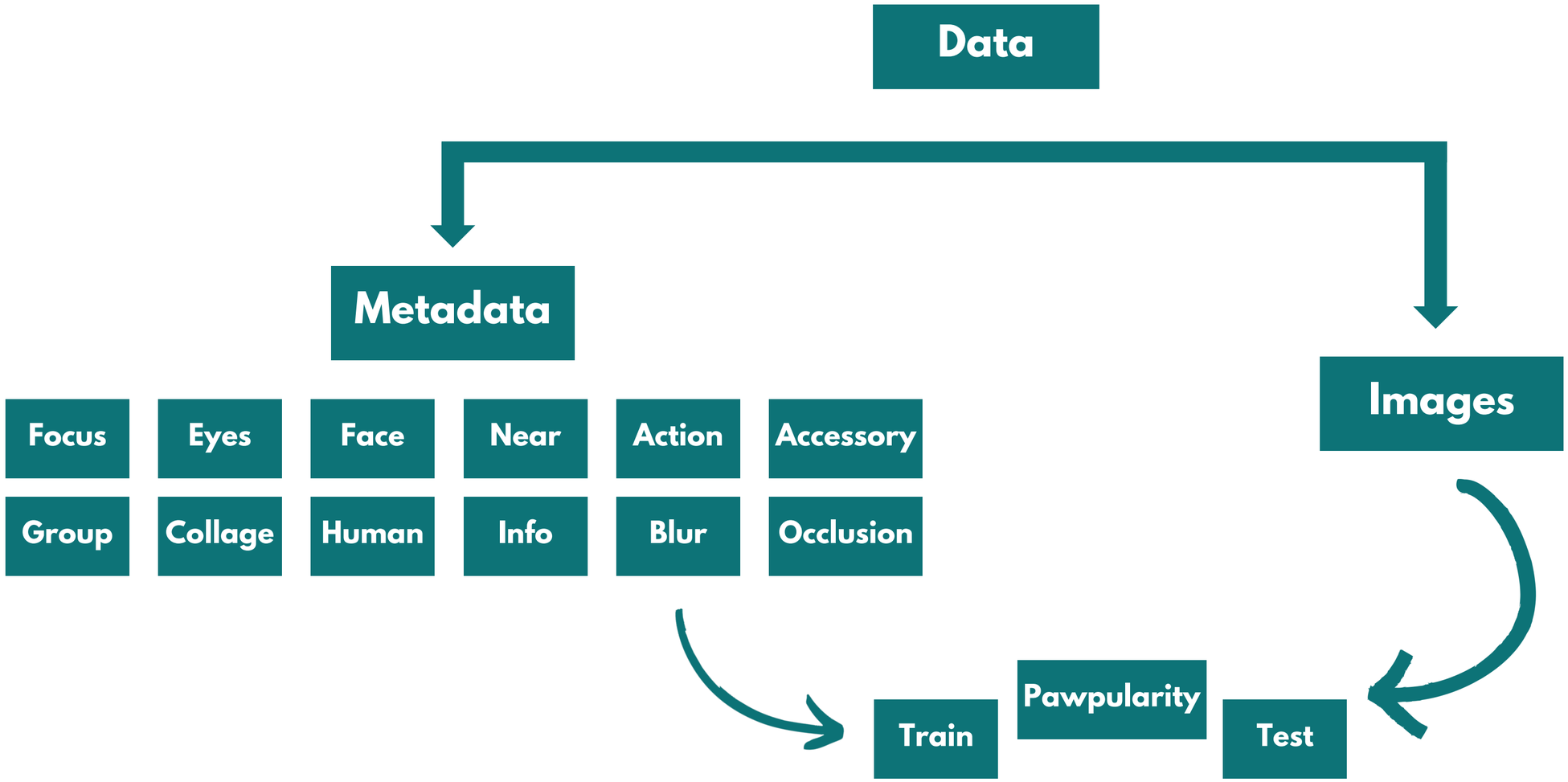
Complication

The existing cuteness ranking meter, while helpful, is still in experimental stage and requires significant enhancement to provide precise recommendations.

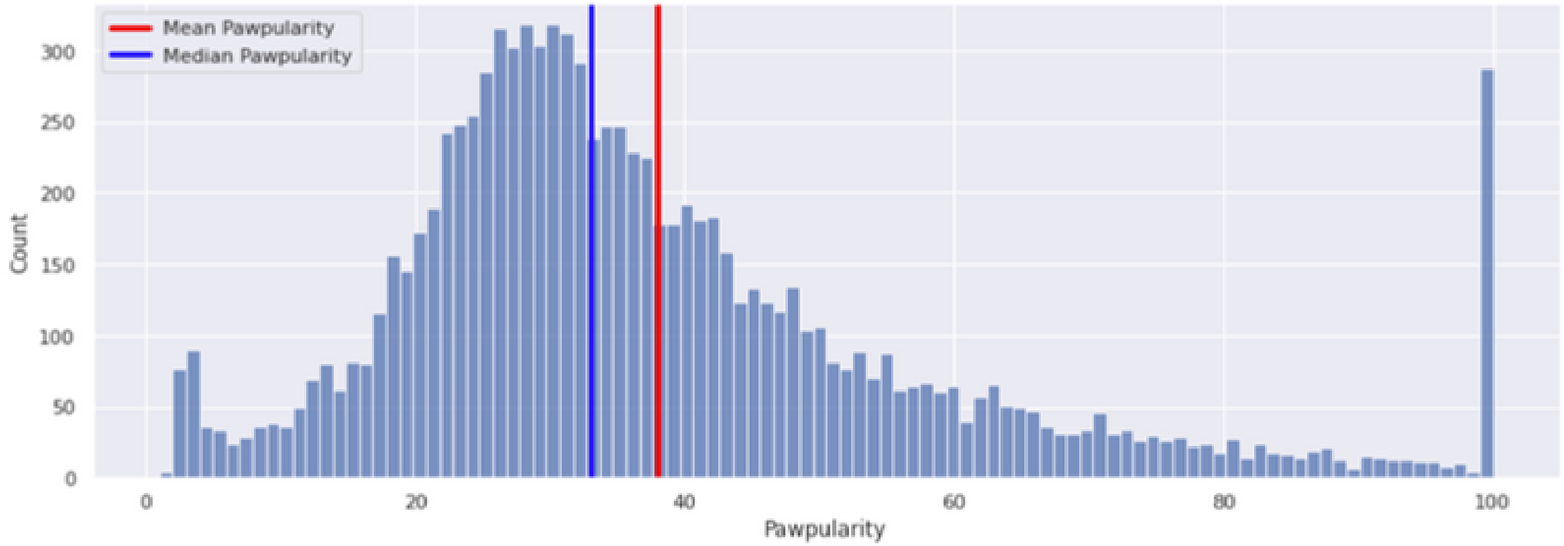
Key Question

How can we enhance the current prediction algorithm by better analyzing the images to predict the “Pawpularity” of pet photos, ensuring that the model provides accurate recommendations to increase adoption rates?

Data structure

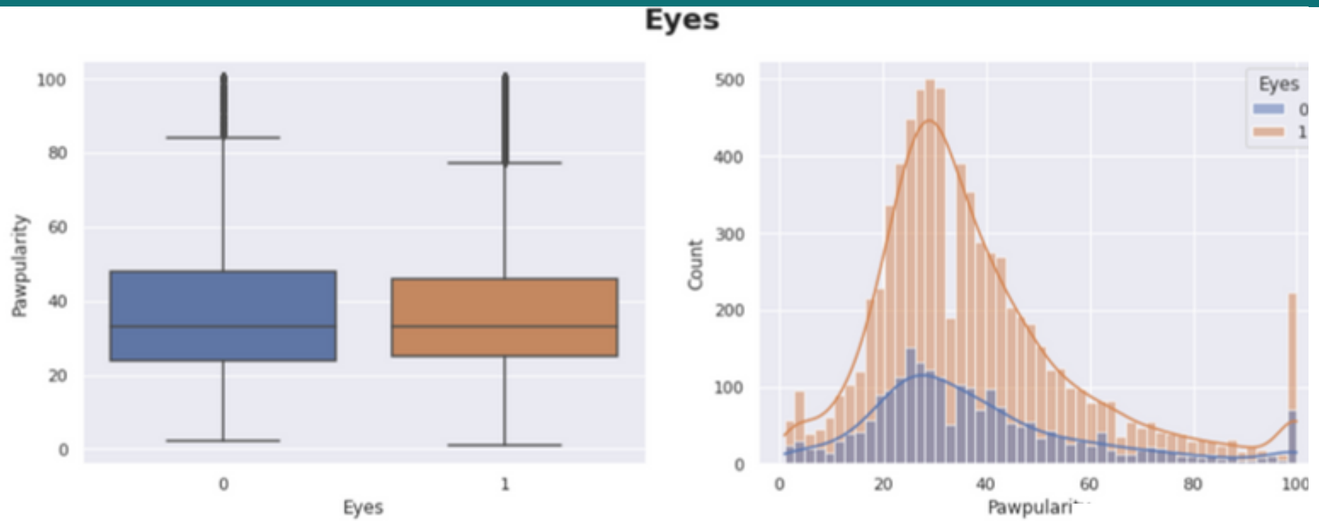


Exploratory Results

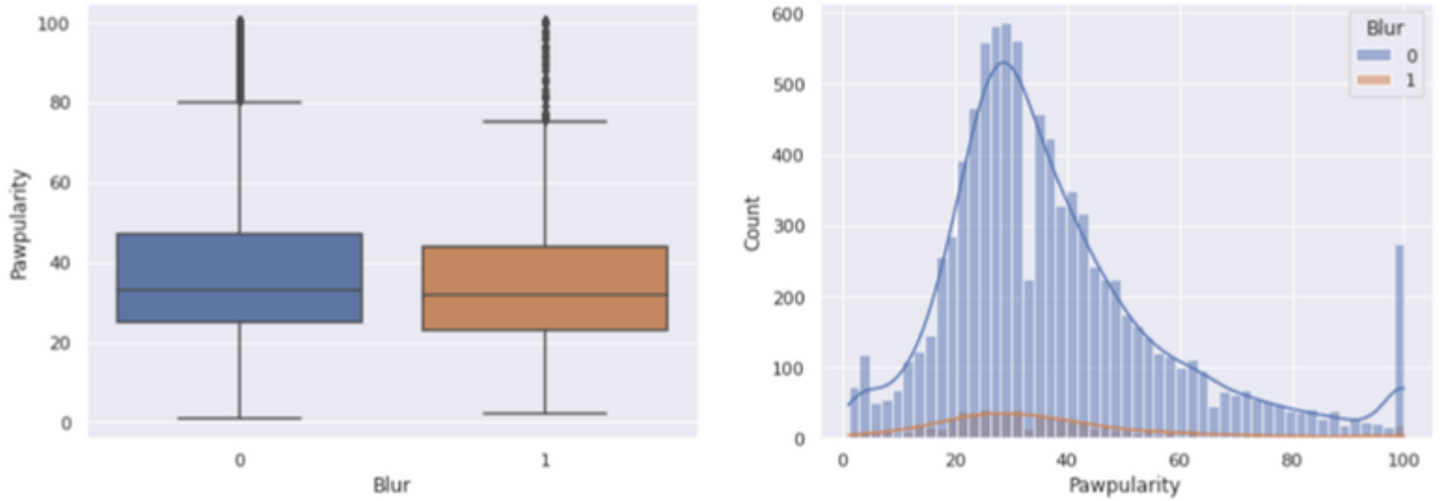


- *The distribution is **right-skewed**, that is there are a number of images with popularity scores significantly higher than the mode.*
- *The **median popularity**, or the middle value, is around 30. This means that half of the images have a popularity score below 30 and half have a score above 30.*
- *There is a **noticeable spike** in the count at around 100 popularity. This could suggest that there are some exceptionally popular images that stand out from the rest*

Exploratory Results

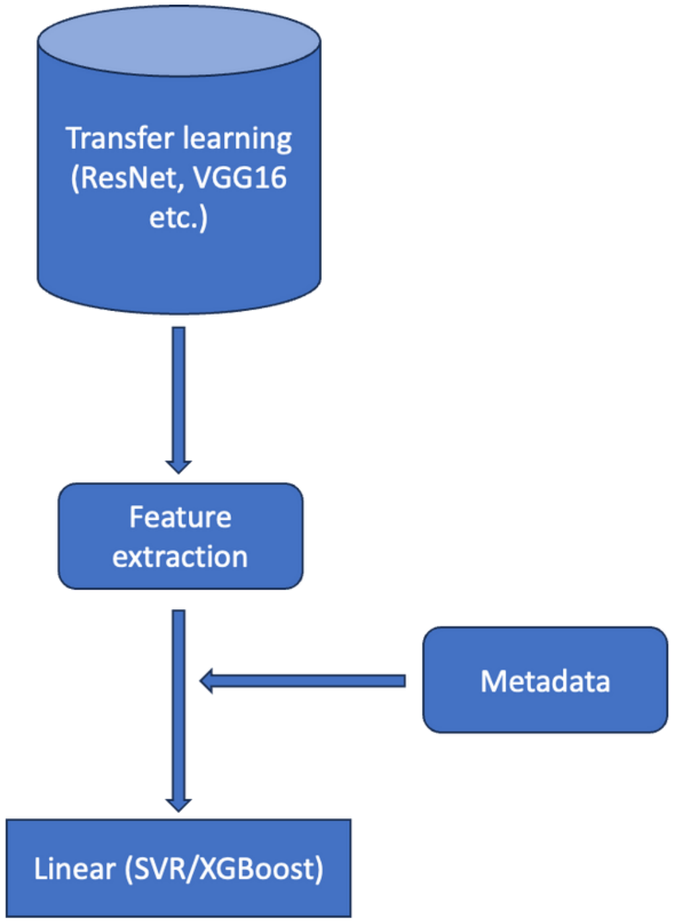


Distribution of Eyes over Pawpularity Score



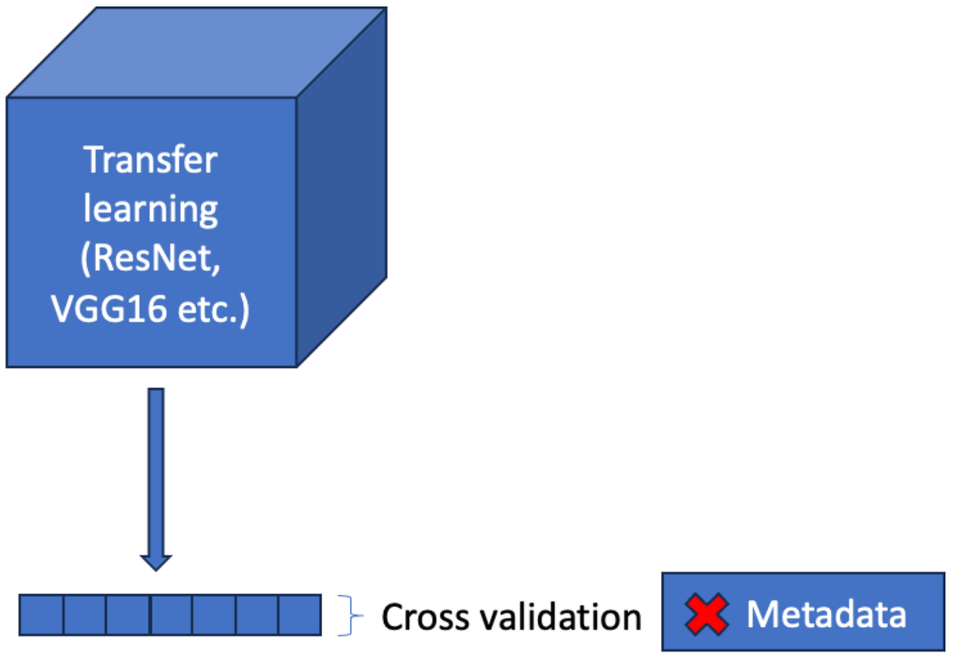
Distribution of Blur over Pawpularity Score

Strategy


Transfer learning (ResNet, VGG16 etc.)
Feature extraction
Metadata
Linear (SVR/XGBoost)

1

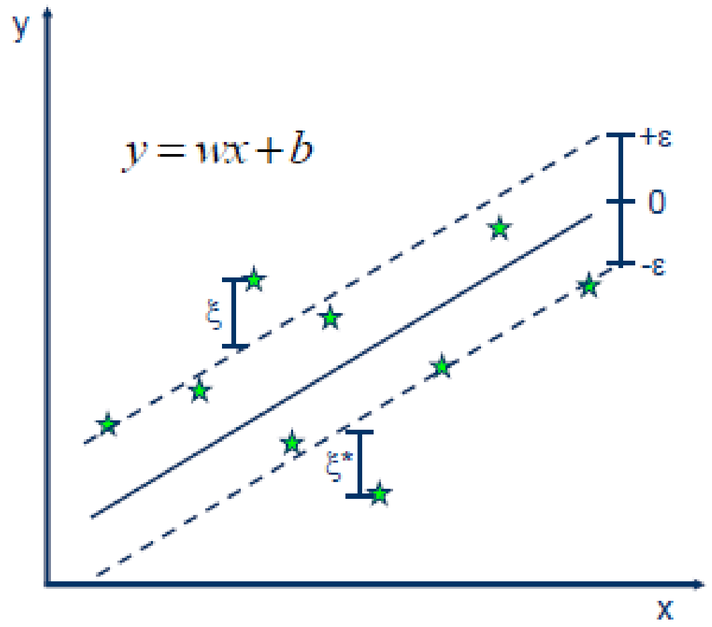
2


Transfer learning (ResNet, VGG16 etc.)
Cross validation
Metadata

Modeling

1. **SupportVR** — Metadata
2. **CNN** — Images
1. **ResNet** — Metadata | Images
1. **EfficientNet** — Metadata | Images
1. **ResNet + EfficientNet + SVR** — Metadata | Images

Support Vector Regression



Linear SVR

- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

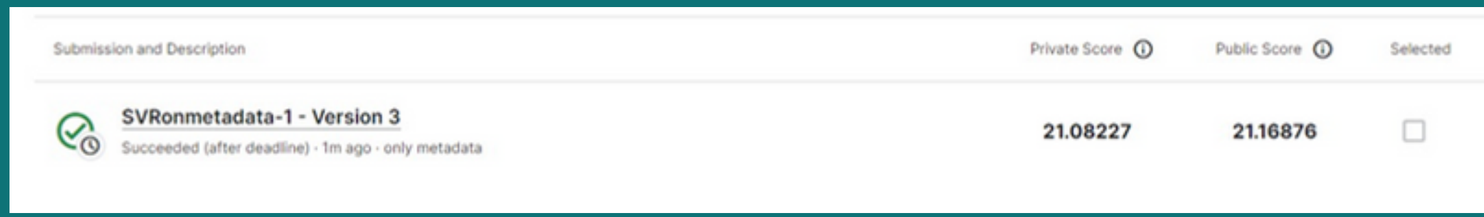
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

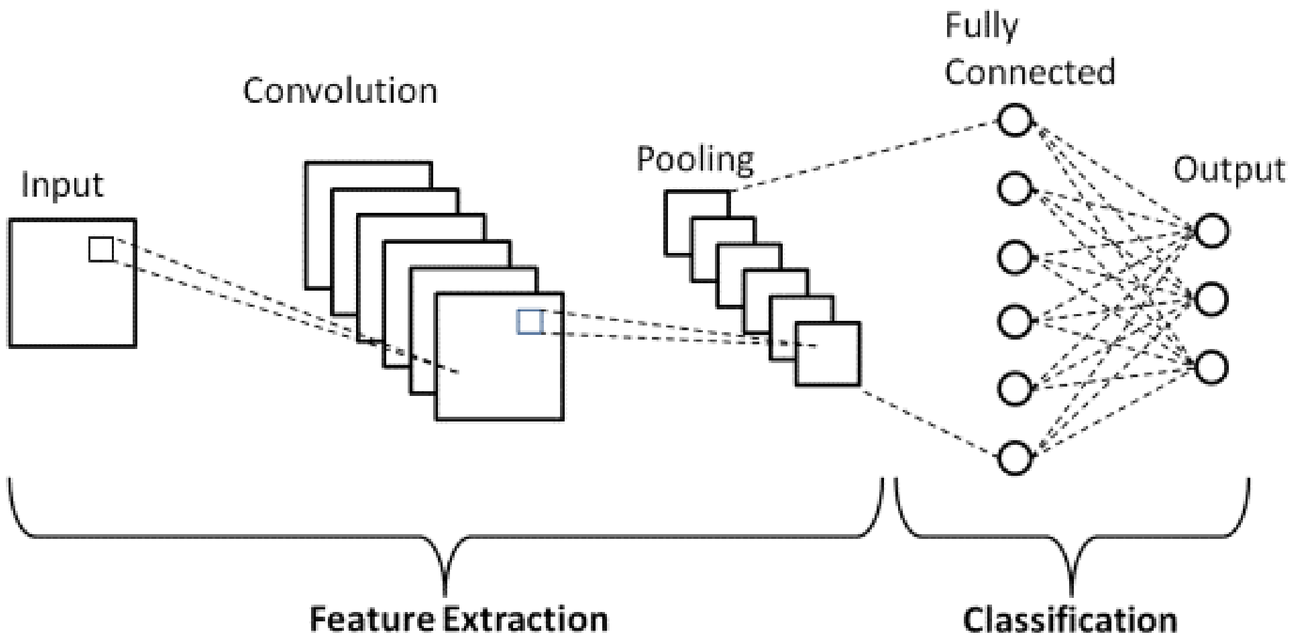
Model 1: SVR on Metadata

```
Xm = metadata_df.drop(columns=['Id', 'Pawpularity'])
Ym = metadata_df["Pawpularity"]
```

```
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error
SVR = SVR()

#Fit the model on the training data
SVR.fit(Xm, Ym)

# Make predictions on the testing data
SVR_predictions = SVR.predict(x_test)

print(SVR_predictions)
```

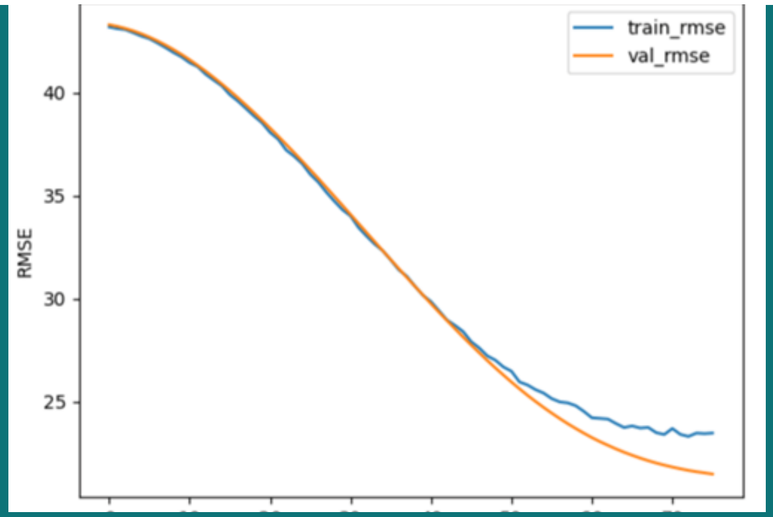


CNN architecture



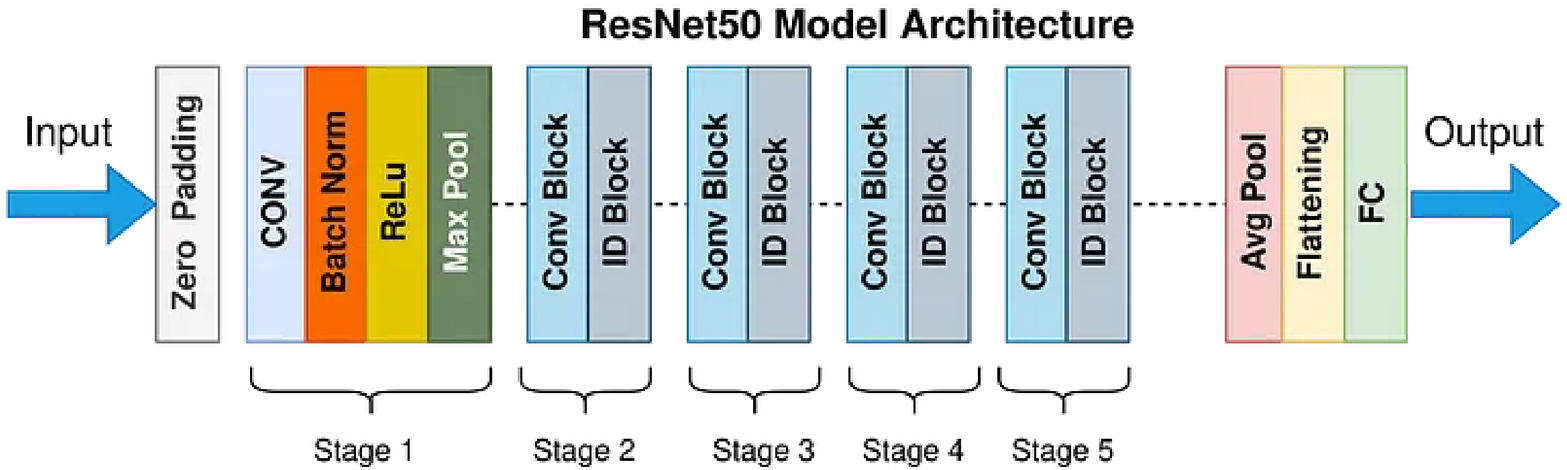
Model 2: CNN on images

```
el = tf.keras.Sequential()

el.add(tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(128,128,3)))
el.add(tf.keras.layers.MaxPool2D((2,2)))

el.add(tf.keras.layers.Conv2D(32,(3,3),activation='relu'))
el.add(tf.keras.layers.MaxPool2D((2,2)))

el.add(tf.keras.layers.Conv2D(64,(3,3),activation='relu'))
el.add(tf.keras.layers.MaxPool2D((2,2)))

el.add(tf.keras.layers.Flatten())
el.add(tf.keras.layers.Dense(64,activation='relu'))
el.add(tf.keras.layers.Dropout(0.1))
el.add(tf.keras.layers.Dense(1,activation='linear'))
el.add(tf.keras.layers.Dropout(0.1))
el.add(tf.keras.layers.Dense(1,activation='linear'))
```



CNN-final - Version 1 Succeeded (after deadline) · 6h ago · Customized CNN		21.58767	21.65214

Resnet architecture



Model 3: Resnet with metadata

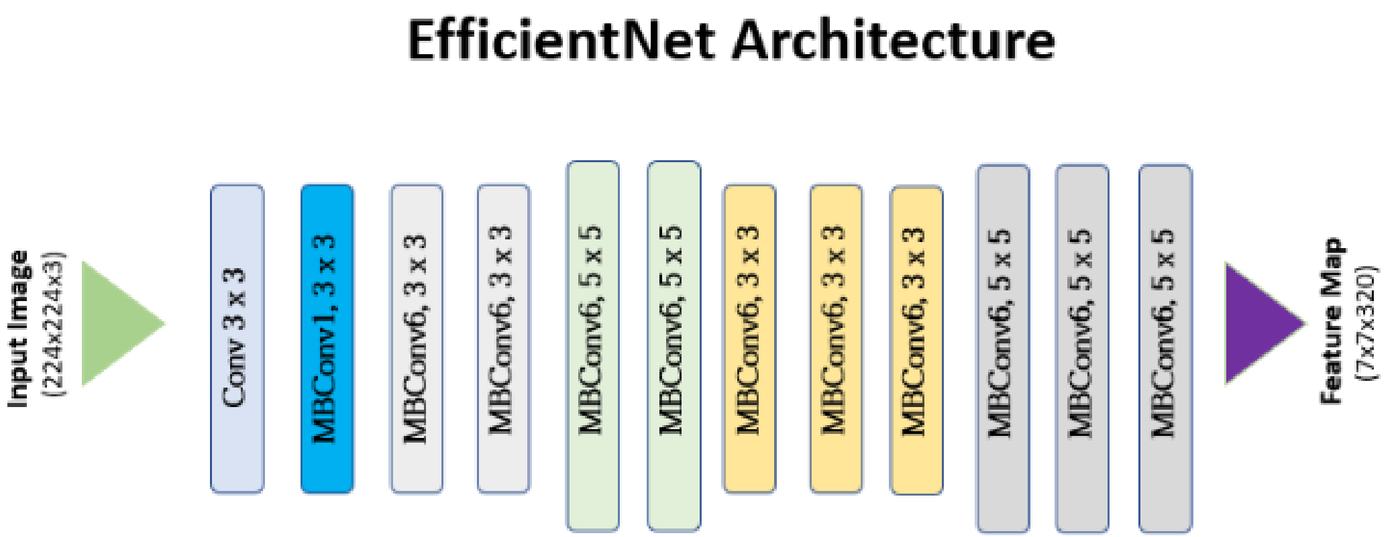
```
tf.keras.backend.clear_session()
models = []
historys = []
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
train_best_fold = True
best_fold = 0
for index, (train_indices, val_indices) in enumerate(kfold.split(train)):
    if train_best_fold and index != best_fold: continue
    x_train = train.loc[train_indices, "file_path"]
    tabular_train = train.loc[train_indices, ["Pawpularity"] + columns]
    x_val= train.loc[val_indices, "file_path"]
    tabular_val = train.loc[val_indices, ["Pawpularity"] + columns]
    checkpoint_path = "model_%d.h5"%(index)
    checkpoint = tf.keras.callbacks.ModelCheckpoint(checkpoint_path, save_best_only=True)
    early_stop = tf.keras.callbacks.EarlyStopping(min_delta=1e-4, patience=10)
    reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(factor=0.3,patience=2, min_lr=1e-7)
    callbacks = [early_stop, checkpoint, reduce_lr]
    optimizer = tf.keras.optimizers.Adam(1e-3)
    train_ds = tf.data.Dataset.from_tensor_slices((x_train, tabular_train)).map(preprocess).shuffle(512).batch(batch_size).cache().prefetch(2)
    val_ds = tf.data.Dataset.from_tensor_slices((x_val, tabular_val)).map(preprocess).batch(batch_size).cache().prefetch(2)
    model = get_model(image_size, columns)
    model.compile(loss = "mse", optimizer = optimizer, metrics = ["mae", rmse, "mape"])
    history = model.fit(train_ds, epochs=300, validation_data=val_ds, callbacks=callbacks, batch_size = 8)
    for metrics in [("loss", "val_loss"), ("mae", "val_mae"), ("mape", "val_mape"), ["lr"]]:
        pd.DataFrame(history.history, columns=metrics).plot()
        plt.show()
    model.load_weights(checkpoint_path)
    historys.append(history)
    models.append(model)
```

Model 4: Resnet model 3 fine tuned

```
def build_tabular_model(inputs):

    x = keras.layers.Dense(12, activation='relu')(inputs)
    x = keras.layers.Dense(64, activation='relu')(x)
    x = keras.layers.Dropout(0.3)(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Dense(128, activation='relu')(x)
    x = keras.layers.Dense(64, activation='relu')(x)
    x = keras.layers.Concatenate()([x, inputs])
    return x
```

ResnetwithSVR - Version 2 Succeeded (after deadline) · 4d ago	**20.27842**	**20.23979**	☐
Resnetwithmetadata2 - Version 13 Succeeded (after deadline) · 4d ago · resnet with metadata as final layer of neural network	**20.50793**	**20.51378**	☐

EfficientNet Architecture



Model 5: Efficientnet

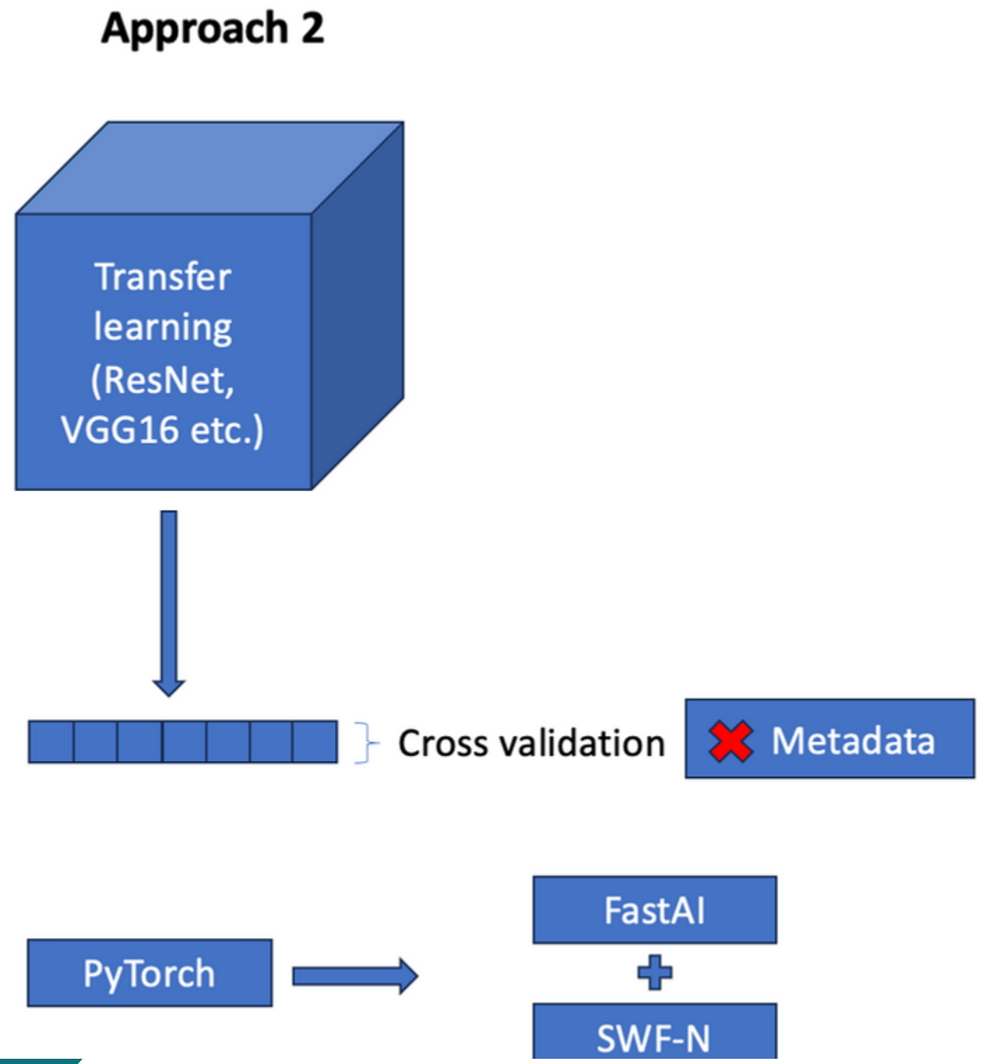
```python
import os
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Concatenate, RandomContrast, RandomRotation
from tensorflow.keras.models import Model
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications import EfficientNetB0

def get_model(image_size, columns):
    image_inputs = tf.keras.Input((image_size, image_size, 3))
    tabular_inputs = tf.keras.Input(len(columns))

    #weights_path = '/kaggle/input/resnet50-weights-h5/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5'
    #if os.path.exists(weights_path):
     #   resnet = ResNet50(include_top=False, weights=weights_path, pooling=None)
    #else:
    #print("Weights file not found. Using 'imagenet' weights.")
    resnet = tf.keras.applications.EfficientNetB0(include_top=False, weights='imagenet', pooling=None)
    image_x = resnet(RandomContrast(factor=0.1)(RandomRotation(factor=0.15)(image_inputs)))
    image_x = tf.keras.layers.GlobalAveragePooling2D()(image_x)

    # Assuming build_tabular_model is a function you have defined to handle tabular data
    tabular_x = build_tabular_model(tabular_inputs)

    x = tf.keras.layers.Concatenate(axis=1)([image_x, tabular_x])
    output = tf.keras.layers.Dense(1)(x)
    model = tf.keras.Model(inputs=[image_inputs, tabular_inputs], outputs=[output])
    return model
```

Python · resnet50_weights.h5, PetFinder.my - Pawpularity Contest

Notebook | Input | Output | Logs | Comments (0) | Settings

Competition Notebook	Run	Best Score
PetFinder.my - Pawpularity Contest	1317.7s - GPU P100	20.27842 V2

Revised Model approach

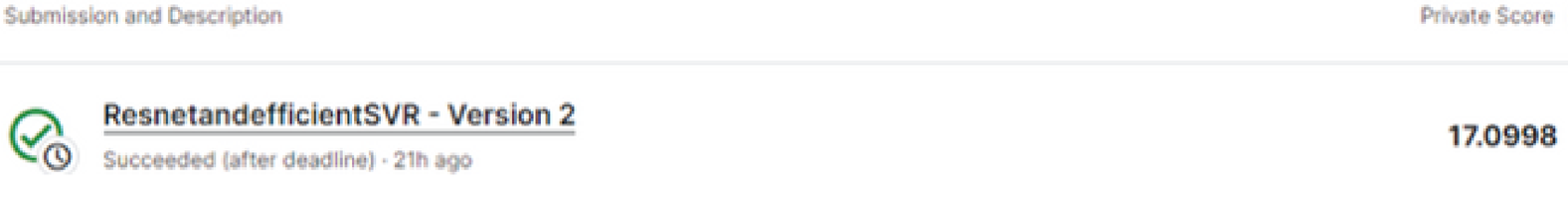


Model 6: ResNet+ EfficietNet + SVR

```
# timm_model_lst = ["../input/timm-pretrained-models/beitv2_large_patch16_224.pkl"]
timm_model_lst = ["../input/timm-pretrained-models/tf_efficientnet_l2_ns_475.pkl",
                  "../input/timm-pretrained-models/beitv2_large_patch16_224.pkl"]
#"../input/timm-pretrained-models/swin_base_patch4_window7_224.pkl"
                  # "../input/timm-pretrained-models/vit_large_r50_s32_384.pkl",
                  # "../input/timm-pretrained-models/vit_large_patch16_384.pkl"]
for m in timm_model_lst:
    print("Extracing features using", m)
    pretrained_model = pickle.load( open( m, "rb" ) ).to('cuda')
    trans_config = resolve_data_config({}, model=pretrained_model)
    default_trans = create_transform(**trans_config)

    test_dataset = PetfinderDataSet(img_id_lst=test_id_lst, transform=default_trans, base_path=test_data_base_path)
    test_data_loader = DataLoader(test_dataset, batch_size=batch_size, num_workers=2, shuffle=False)

    with torch.no_grad():
        emb = [pretrained_model(data.to('cuda')).cpu().numpy() for data in test_data_loader]
    EMB_full = np.concatenate(emb, 0)

    model_name = m.split("/")[-1].split(".")[0]
    pickle.dump( EMB_full, open( "./testset_features/"+model_name+"_features.pkl", "wb" ) )

    del pretrained_model, EMB_full
    torch.cuda.empty_cache()
    gc.collect()
    print("Done!\n")
```

Model 6: ResNet+ EfficietNet + SVR

```python
# Fixed setting
models = ["RN50x16",
          "RN50x4",
          "ViT-B-16",
          "ViT-B-32",
          "tf_efficientnet_l2_ns_475"]
model_old_hill_1 = ["RN50x16",
                    "beitv2_large_patch16_224",
                    "beitv2_large_patch16_224"] # * 1.032
model_old_hill_2 = ["ViT-B-16",
                    "beitv2_large_patch16_224"] # * 1.032
models_man_2 = ["RN50x16",
                "RN50x4",
                "ViT-B-16",
                "ViT-B-32",
                "tf_efficientnet_l2_ns_475",
                "beitv2_large_patch16_224"] # loss: 17.296  * 1.030
models_man_5 = ["RN50x16",
          "RN50x4",
          "ViT-B-16",
          "ViT-B-32",
          "beitv2_large_patch16_224",
          "swin_base_patch4_window7_224",
          "tf_efficientnet_l2_ns"] # loss: 17.257  * 1.029
# "beitv2_large_patch16_224"
#           "vit_large_patch16_384",
#           "vit_large_r50_s32_384"
# models = ["beitv2_large_patch16_224",
#           "RN50x16"]
models = models_man_2
EMB_test = None
for i in range(len(models)):
    if i == 0:
        EMB_test = pickle.load( open( "./testset_features/"+models[0]+"_features.pkl", "rb" ) )
    else:
        EMB_test = np.concatenate((EMB_test, pickle.load( open( "./testset_features/"+models[i]+"_features.pkl", "rb" ) )), axis=1)
```



Performance

Model	RMSE
SVR	21.08
CNN	21.59
ResNet	20.28
EfficientNet	20.28
ResNet + EfficientNet+ SVR	17.08

Insights

Positive Contribution of Metadata

CNN Limitations

Transfer Learning Superiority

Multiple Transfer Learning Models

Effective Hyperparameter Tuning

Significance of Data Augmentation

THANK YOU