

Happywhale - Whale and Dolphin Identification

Identify whales and dolphins by unique characteristics

Javier Ferrero Micó



IDAL

Intelligent
Data
Analysis
Laboratory

Descripción del problema

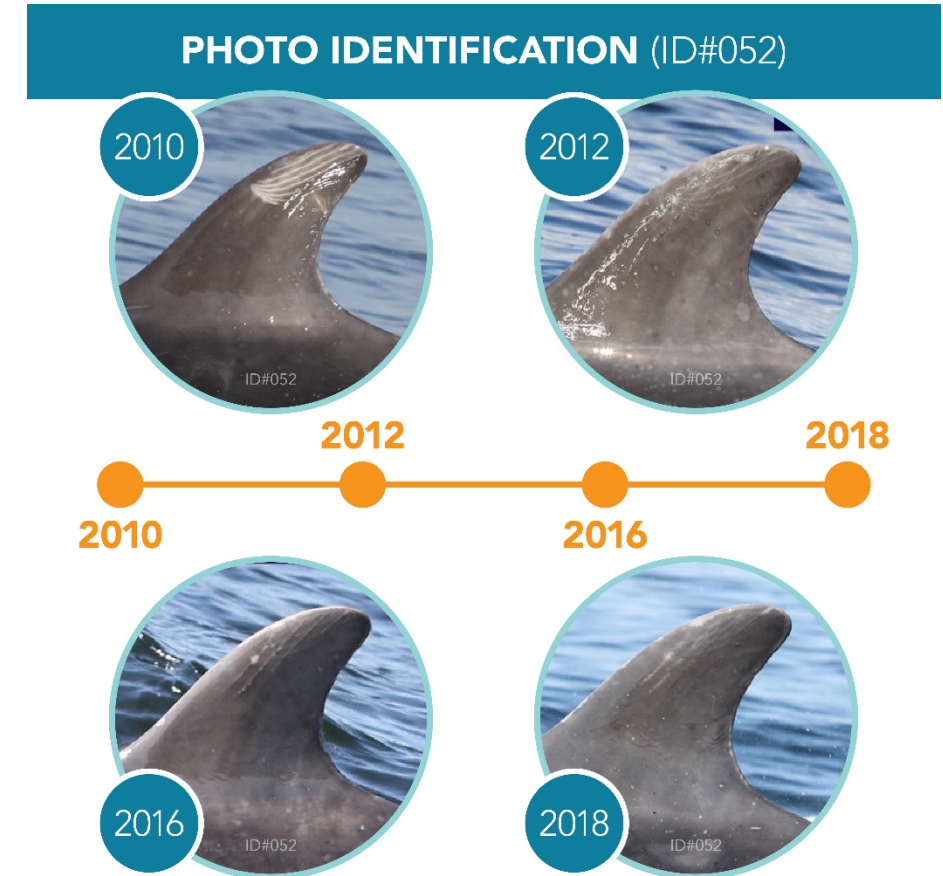
Competición Kaggle

Utilizamos las huellas dactilares y el reconocimiento facial para identificar a las personas, pero ¿podemos utilizar enfoques similares con los animales?

Los investigadores rastrean manualmente la vida marina por la forma y las marcas de sus colas, aletas dorsales, cabezas y otras partes del cuerpo. Esta es una poderosa herramienta que permite seguir a los animales a lo largo del tiempo y evaluar el estado y las tendencias de la población.

La competición pide automatizar dicha tarea, de forma eficiente, mediante técnicas de visión por computador y Deep learning.

<https://www.kaggle.com/competitions/happy-whale-and-dolphin>



Descripción del problema

Competición Kaggle

El problema se puede dividir en varias fases:

1. Filtrado de todas las fotografías en las distintas especies
2. Dentro de cada especie, segmentación de la zona que es aleta y la que es agua
3. Dentro de cada aleta, detección de las marcas características de cada espécimen en particular

Problema de clasificación multiclass

image	species	individual_id
51033 unique values	bottlenose_dolphin 19% beluga 15% Other (33926) 66%	15587 unique values
00021adfb725ed.jpg	melon_headed_whale	cadddb1636b9
000562241d384d.jpg	humpback_whale	1a71fbb72250
0007c33415ce37.jpg	false_killer_whale	60008f293a2b
0007d9bca26a99.jpg	bottlenose_dolphin	4b00fe572063
00087baf5cef7a.jpg	humpback_whale	8e5253662392
000a8f2d5c316a.jpg	bottlenose_dolphin	b9907151f66e
000be9acf46619.jpg	beluga	afb9b3978217
000bef247c7a42.jpg	humpback_whale	444d8894ccc8
000c3d63069748.jpg	beluga	df94b15285b9
000c476c11bad5.jpg	bottlenose_dolphin	b11b2404c7e3
001001f099519f.jpg	minke_whale	19fbb960f07d
00103cbe9d25ce.jpg	fin_whale	180c0ab04dcd
00144776eb476d.jpg	bottlenose_dolphin	b9907151f66e
00167e8375c967.jpg	beluga	0ad50d0d9b06
00177f3c614d1e.jpg	bottlenose_dolphin	812be36c2aef
001782740d750.jpg	bottlenose_dolphin	41527040d450

Descripción de los datos

Competición Kaggle

- Dataset descompensado, existen hasta 30 clases distintas de ballena/delfín en el dataset. 37,5 GB de datos iniciales. Se necesita diezmar selectivamente
 - Se han desestimado las clases que no tienen suficientes datos (Solo 13 clases tienen más de 1000 imágenes)
 - Para compensar las clases, se han diezmando aquellas con sobrerrepresentación
 - Se ha aplicado data augmentation para incrementar la variabilidad de todas las clases por igual
 - Se ha normalizado los valores de los datos entre {0,1}
- Para la primera fase del problema, desestimaremos los datos del individual_id
- Los tamaños de las imágenes son muy variables – se requiere resize – 224 x 224 x 3
- No son imágenes normalizadas. Para las siguientes fases, se requiere una normalización de las mismas, para centrar el objeto de estudio y eliminar ruido gaussiano

	Cantidad de datos
Trainset	8247 imágenes de 13 clases distintas
Validation set	6922 imágenes de 13 clases distintas
Test set	2042 imágenes de 13 clases distintas

```
train_datagen = ImageDataGenerator(  
    rescale = 1./255,  
    rotation_range=40,  
    width_shift_range=0.2,  
    height_shift_range=0.2,  
    fill_mode='mirror',  
    shear_range=0.2,  
    zoom_range=0.2,  
    horizontal_flip=True)
```


Descripción de los datos - muestras

Competición Kaggle



Modelo de deeplearning - caracterización

Competición Kaggle

- Arquitectura elegida: MobileNetV2 modificando top.
 - Width multiplier $\alpha = 1, 0.35$. Loss = categorical_crossentropy
 - Optimizer = Adam, learning rate por defecto de $1e-3$.
 - Batch size = 32
 - Aplicamos Feature extraction
 - Pesos inicialmente cargados de imagenet
 - Se entrenan todas las capas
- Modelo entrenado mediante una GPU Local NVIDIA GeForce RTX 2070
- Top añadido mediante la API funcional de Keras.
 - Una primera capa de average pooling pooling para pasar a 1 dimension, tras las convoluciones de MobileNet.
 - Finalmente unas capas densas para categorizar las últimas características y enlazarlas a las clases deseadas,
 - Intercalando regularizadores (dropout) para reducir el overfitting y estresar al máximo el proceso de backpropagation, “congelando” una parte de los pesos

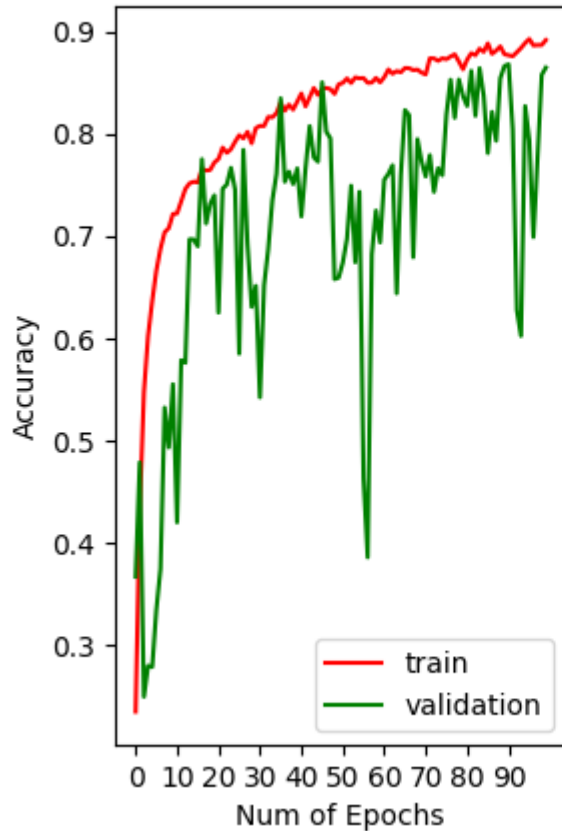
```
aux0 = model.output
aux1 = GlobalAveragePooling2D()(aux0)
aux2 = Dense(512, activation='relu')(aux1)
drop2 = Dropout(0.7)(aux2)
aux3 = Dense(256, activation='relu')(drop2)
drop3 = Dropout(0.6)(aux3)
aux4 = Dense(128, activation='relu')(drop3)
drop4 = Dropout(0.5)(aux4)
aux5 = Dense(64, activation='relu')(drop4)
drop5 = Dropout(0.3)(aux5)
aux6 = Dense(len(classes), activation='softmax')(drop5)
new_model = Model(model.input, aux6)
```

Resultados - métricas

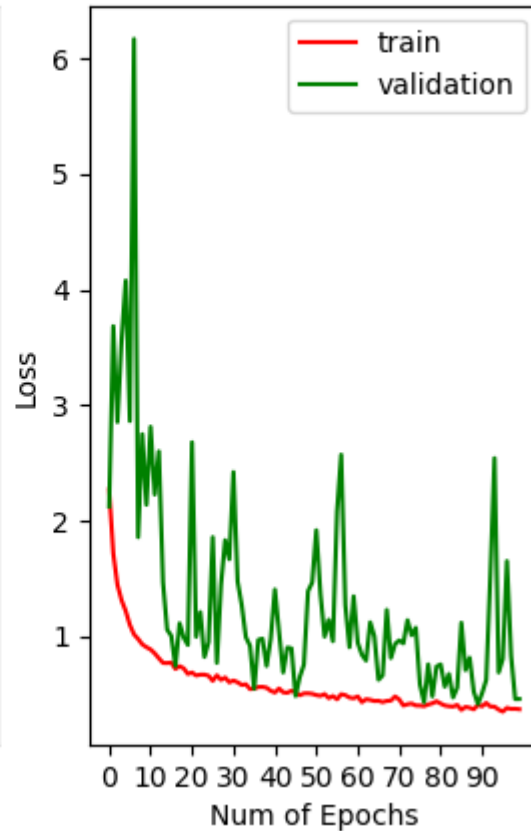
Competición Kaggle

100 épocas
258 steps por época

Training Accuracy vs Validation Accuracy



Training Loss vs Validation Loss



	Loss	Accuracy
Train	0.3690	0.8915
Validation	0.4583	0.8646
Test	0.4875	0.8594

Test MSE	0.0159
AUC	0.9840

Resultados - métricas

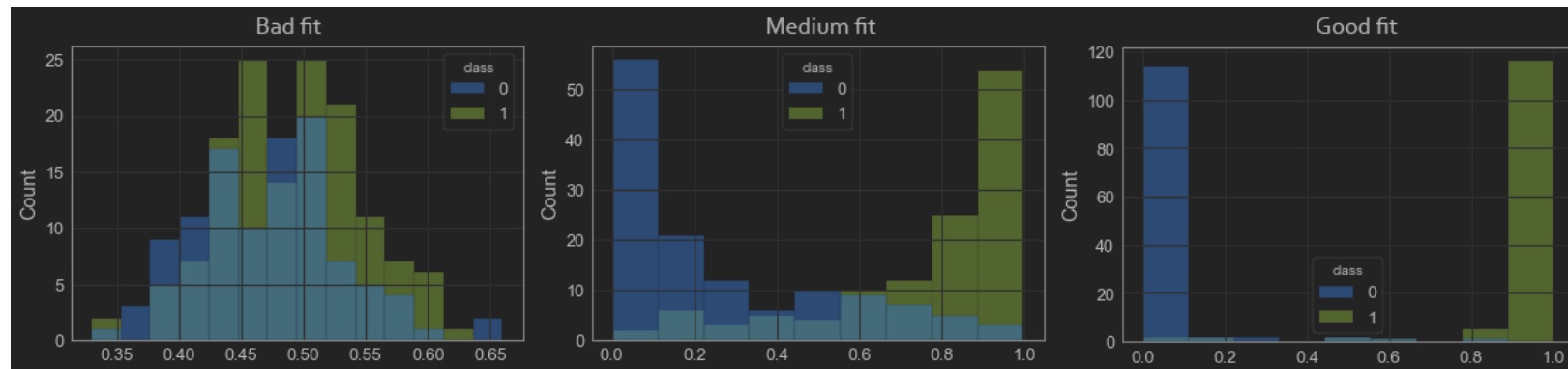
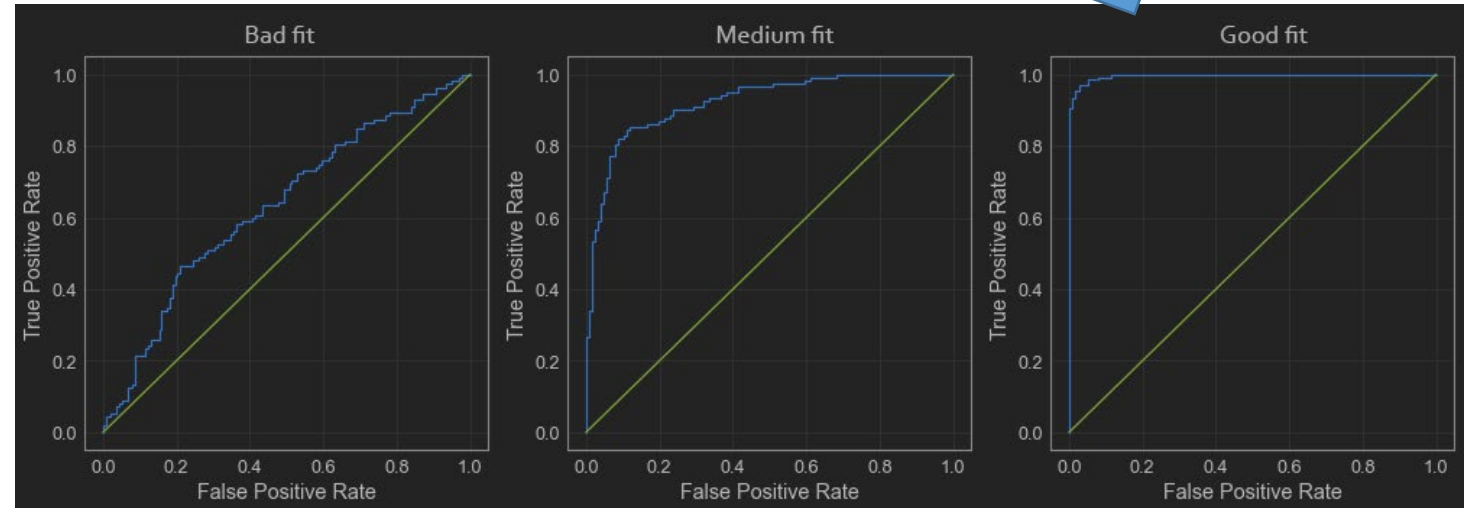
Competición Kaggle

Teniendo en cuenta el valor de AUC cercano a 1, se esperaría



ROC Multiclass – One vs Rest

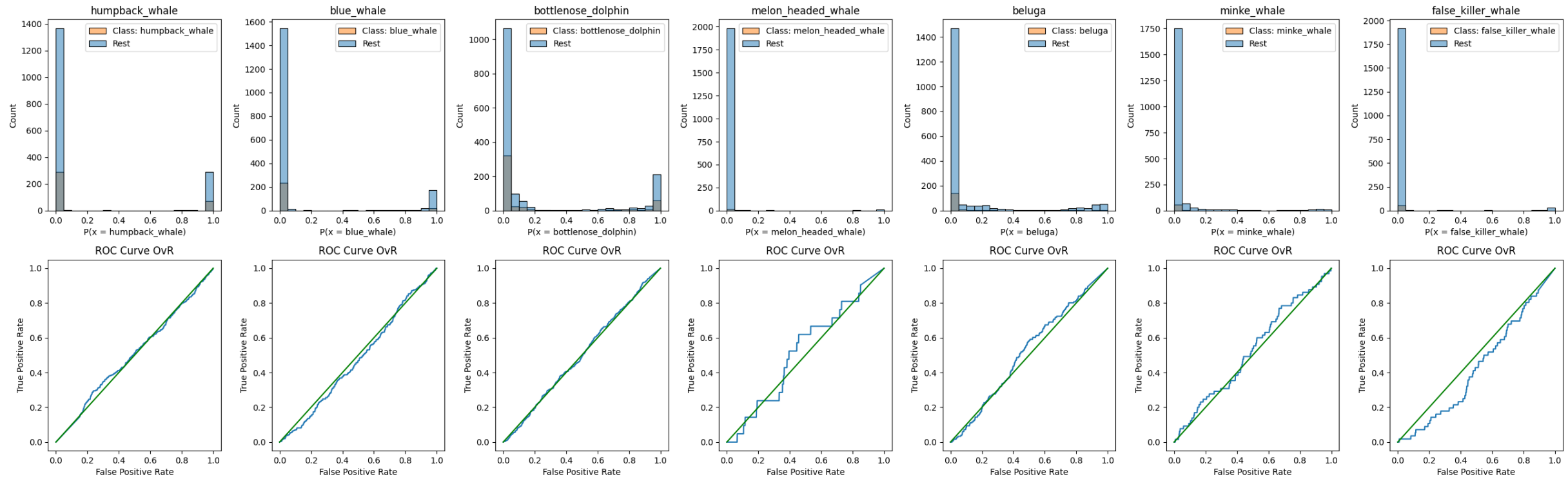
- Iterar sobre todas las clases
- Preparar un marco de datos auxiliar utilizando una clase como "1" y las otras como "0"
- Traza los histogramas de las distribuciones de las clases
- Traza la curva ROC para cada caso
- Calcula el AUC para esa clase específica



Resultados - métricas

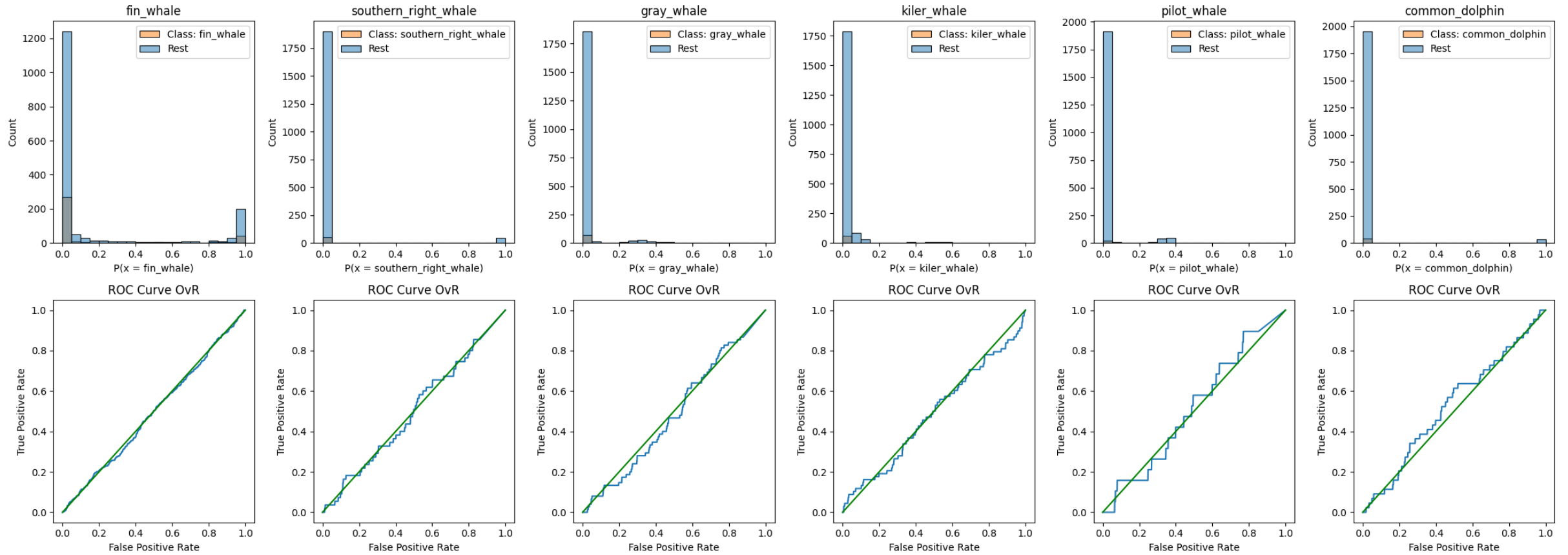
Competición Kaggle

No se ha conseguido separar correctamente el OvR – se debe revisar el pipeline



Resultados - métricas

Competición Kaggle



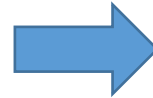
Resultados - métricas

Competición Kaggle

```
for i in range(len(classes)):
    # Gets the class
    c = classes[i]
    print(str(c))
    # Prepares an auxiliar dataframe to help with the plots
    df_aux = pd.DataFrame(np.zeros(shape=(2042, 2)), columns=['class', 'prob'])
    #df_aux = y_test.copy()
    df_aux['class'] = [1 if y == i else 0 for y in y_test]
    df_aux['prob'] = y_proba[:, i]
    df_aux = df_aux.reset_index(drop=True)
```



```
ax_bottom = plt.subplot(4, 7, i + 8)
tpr, fpr = get_all_roc_coordinates(df_aux['class'], df_aux['prob'])
plot_roc_curve(tpr, fpr, scatter=False, ax=ax_bottom)
ax_bottom.set_title("ROC Curve OvR")
```



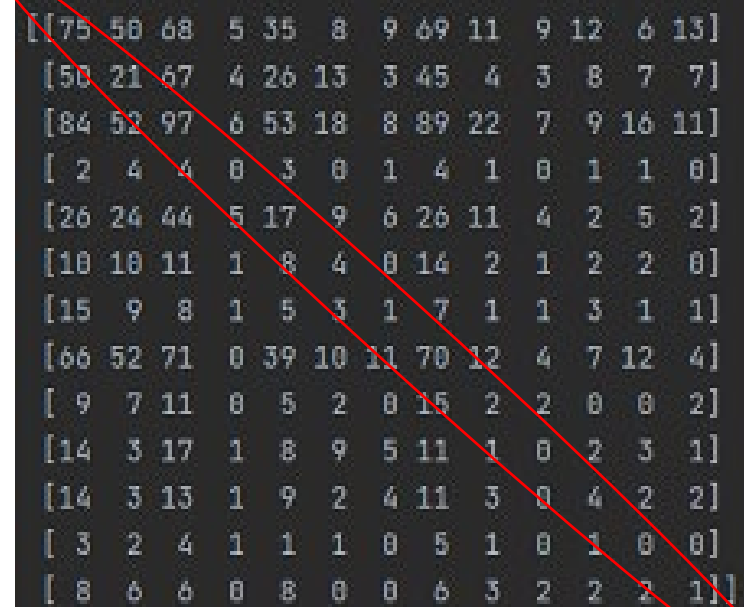
```
tpr_list = [0]
fpr_list = [0]
for i in range(len(y_proba)):
    threshold = y_proba[i]
    y_pred = y_proba >= threshold
    tpr, fpr = calculate_tpr_fpr(y_real, y_pred)
    tpr_list.append(tpr)
    fpr_list.append(fpr)
return tpr_list, fpr_list
```

Resultados - métricas

Competición Kaggle

	precision	recall	f1-score	support
humpback_whale	0.20	0.20	0.20	370
blue_whale	0.09	0.08	0.08	258
bottlenose_dolphin	0.23	0.21	0.22	472
melon_headed_whale	0.00	0.00	0.00	21
beluga	0.08	0.09	0.09	181
minke_whale	0.05	0.06	0.06	65
false_killer_whale	0.02	0.02	0.02	56
fin_whale	0.19	0.20	0.19	358
southern_right_whale	0.03	0.04	0.03	55
gray_whale	0.00	0.00	0.00	75
kiler_whale	0.08	0.06	0.07	68
pilot_whale	0.00	0.00	0.00	19
common_dolphin	0.02	0.02	0.02	44
accuracy			0.14	2042
macro avg	0.08	0.08	0.07	2042
weighted avg	0.15	0.14	0.14	2042

Observamos una elevada tasa de Falsos Positivos, y una tasa muy baja de Verdaderos positivos, lo que arroja unas ROC deficientes



```
[ [75 50 68 5 35 8 9 69 11 9 12 6 13]
  [50 21 67 4 26 13 3 45 4 3 8 7 7]
  [84 52 97 6 53 18 8 89 22 7 9 16 11]
  [ 2 4 4 0 3 0 1 4 1 0 1 1 0]
  [26 24 44 5 17 9 6 26 11 4 2 5 2]
  [10 10 11 1 8 4 0 14 2 1 2 2 0]
  [15 9 8 1 5 3 1 7 1 1 3 1 1]
  [66 52 71 0 39 10 11 70 12 4 7 12 4]
  [ 9 7 11 0 5 2 0 15 2 2 0 0 2]
  [14 3 17 1 8 9 5 11 1 0 2 3 1]
  [14 3 13 1 9 2 4 11 3 0 4 2 2]
  [ 3 2 4 1 1 1 0 5 1 0 1 0 0]
  [ 8 6 6 0 8 0 0 6 3 2 2 2 1]]
```

Mejoras

Competición Kaggle

Por restricciones de recursos y tiempo, no se han podido probar las siguientes tareas, que habrían sido de interés.

- Incorporar otras arquitecturas como Inception o ResNets
- Comparar el efecto del averagePooling2D vs Flatten para este problema en particular
- Utilizar otros ratios de dropout para optimizar la mejor configuración
- Analizar una configuración de top más profunda, para buscar mejores resultados que los obtenidos actualmente

Comprobación de métricas:

- Revisión del pipeline para comprobar que la generación de las curvas ROC está correcto
- Revisar que la distribución del test set es representativa y equiparable a las utilizadas en train y validación – revisión de las matrices de confusión
- Revisar mediante muestreo, sobre imágenes tipo el porqué de los resultados tan desviados de lo esperado

Correcciones tras la presentación

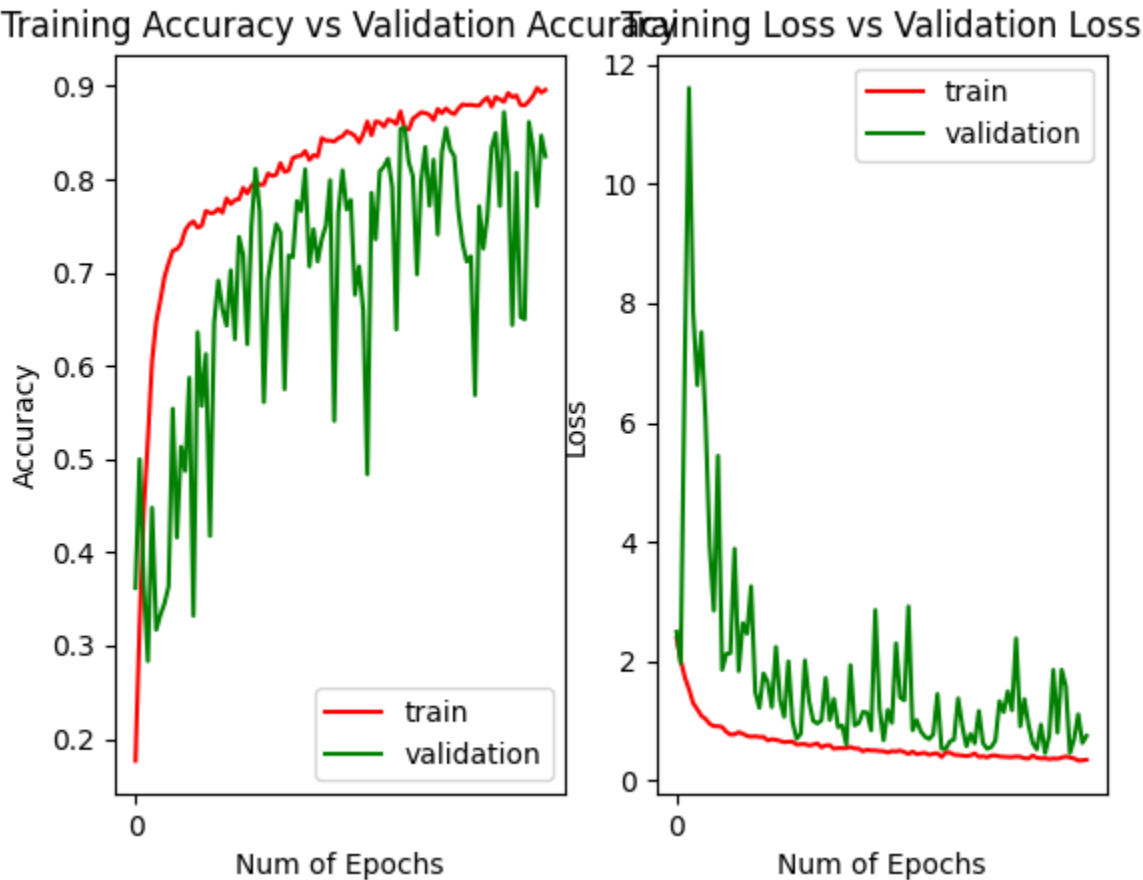
```
# Create the dataset generators
train_generator = train_datagen.flow_from_dataframe(train_set, x_col = 'image', y_col = 'species', target_size = (224, 224), class_mode = class_mode, batch_size = batch_size, shuffle = True)
val_generator = val_datagen.flow_from_dataframe(val_set, x_col = 'image', y_col = 'species', target_size = (224, 224), class_mode = class_mode, batch_size = batch_size, shuffle = False)
test_generator = test_datagen.flow_from_dataframe(test_set, x_col = 'image', y_col = 'species', target_size = (224, 224), class_mode = class_mode, batch_size = batch_size, shuffle = False)
return train_generator, val_generator, test_generator, classes, test_set
```

El test generator tenía un error de composición, al tener shuffle = True, cosa que destruye la relación entre las y_real y las y_preds. Por esto las curvas ROC y la matriz de confusions no funcionaba. Se ha reentrenado el modelo (misma configuración) 100 épocas y estos son los nuevos resultados.

Resultados - métricas

Competición Kaggle

100 épocas
258 steps por época



	Loss	Accuracy
Train	0.3515	0.8960
Validation	0.7528	0.8247
Test	0.8146	0.8002

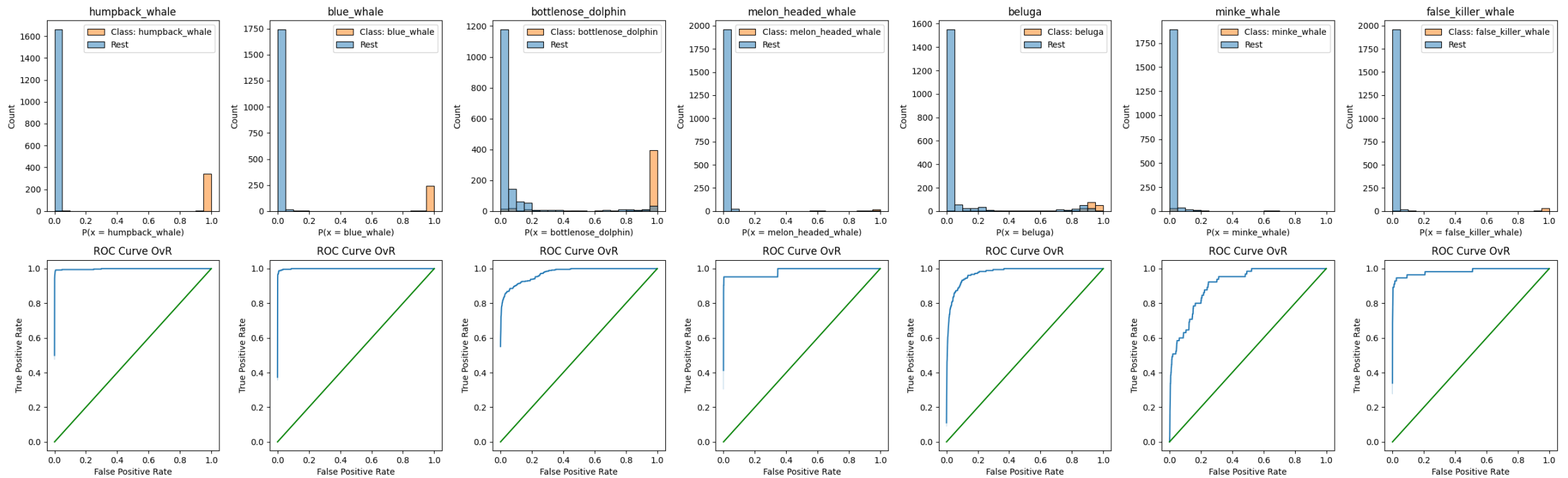
Test MSE	0.0237
AUC	0.9647

Resultados - métricas

Competición Kaggle

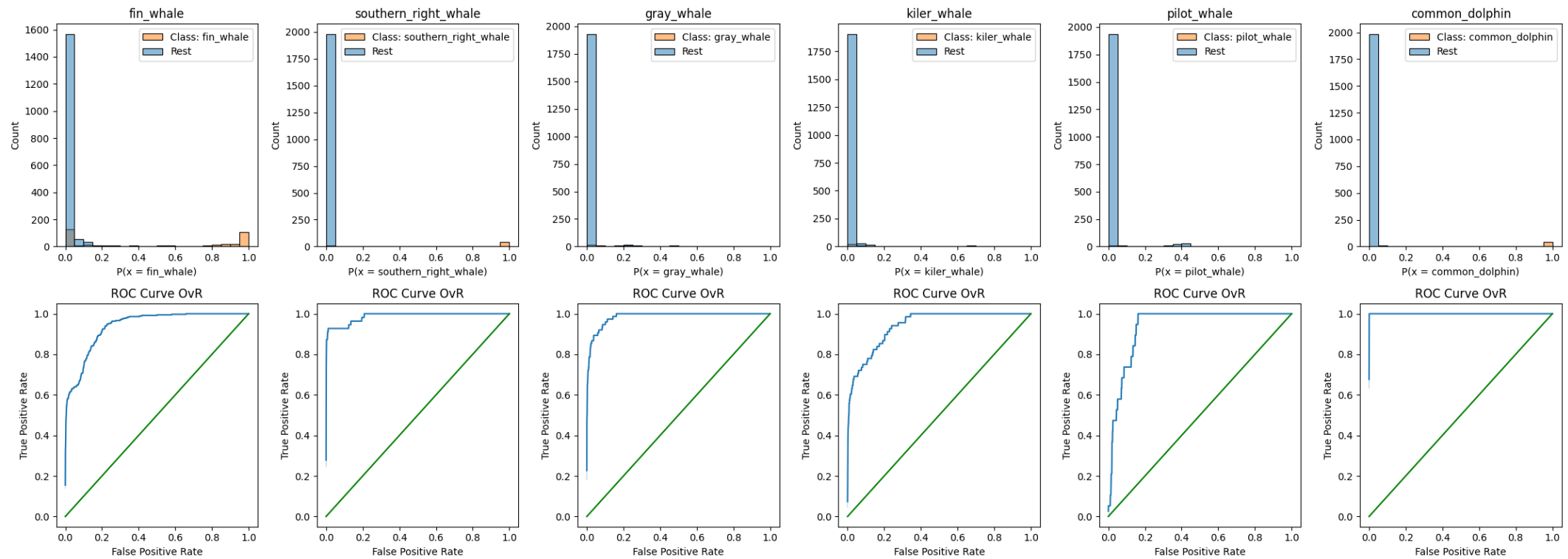
ROC correctas. Alineadas con el valor de AUC obtenido para test.

Todas las clases están bien separadas y existe poco overlay, tal y como se ve en los histogramas.



Resultados - métricas

Competición Kaggle



Resultados - métricas

Competición Kaggle

	precision	recall	f1-score	support
humpback_whale	0.99	0.97	0.98	370
blue_whale	0.95	0.98	0.97	258
bottlenose_dolphin	0.80	0.89	0.84	472
melon_headed_whale	0.40	0.95	0.56	21
beluga	0.51	0.90	0.65	181
minke_whale	0.66	0.29	0.40	65
false_killer_whale	0.91	0.86	0.88	56
fin_whale	0.93	0.54	0.69	358
southern_right_whale	0.90	0.85	0.88	55
gray_whale	1.00	0.35	0.51	75
killer_whale	0.80	0.49	0.61	68
pilot_whale	0.16	0.47	0.23	19
common_dolphin	0.90	1.00	0.95	44
accuracy			0.80	2042
macro avg	0.76	0.73	0.70	2042
weighted avg	0.85	0.80	0.80	2042

Observamos una matriz de confusión casi diagonal, con pocos falsos positivos.

El test set sigue siendo descompensado, fruto de la descompensación generalizada en el dataset.

[illegible]

Muchas gracias por su atención