best_classifiers

September 10, 2025

```
[1]: %load_ext autoreload
     %autoreload 2

     import ipynbname
     import torch
     import os
     import warnings
     import numpy as np
     from xgboost import XGBClassifier
     from libraries.classifier_utils import *
     from libraries.embeddings_utils import *
     from libraries.EmbeddingNet import *
     from libraries.retrieval_utils import *


     project_dir = f"{os.getcwd().
       ↪split('SIDS_revelation_project')[0]}SIDS_revelation_project/"
     image_dataset_path = f"{project_dir}datasets/onback_onstomach_v3"
     model_path_fd = f"{project_dir}/models/4.fd_weights/best.pt"
     model_path_pe = f"{project_dir}/models/2.pe_weights/best.pt"

     if torch.cuda.is_available():
         device = torch.device("cuda")
     elif torch.backends.mps.is_available():
         device = torch.device("mps")
     else:
         device = torch.device("cpu")
     print(f"Using device: {device}")
```

/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.

```
        warnings.warn(

    Using device: cpu
```

```
[2]: emb_builder = EmbeddingBuilder(model_path_fd, image_dataset_path,␣
     ↪"load",weights_path_pe=model_path_pe)
```

```
    Extracting dataset info from .coco.json
    file:-------------------------------------------
    Dataset contains 4158 valid samples, and labels are {'baby_on_back': 1,
    'baby_on_stomach': 2}
    ----------------------------------------------------------------------------
    ----------

    Loading features from
    .csv---------------------------------------------------------------
    Features loaded succesfully, in particular there are 4158 files in the dataset
    ----------------------------------------------------------------------------
    ----------

    Embedding builder initialized
    successfully----------------------------------------------
    Face detection model: 4 (YOLOv8)
    Dataset: /home/terra/Desktop/unimore/AI_engineering/SIDS_revelation_project/data
    sets/onback_onstomach_v3
    Dataset dimension: 4158
    Dataset labels: {'baby_safe': 0, 'baby_unsafe': 1}
    ----------------------------------------------------------------------------
    ----------
```

0.1 Best model with approach Features selection: xgbc

Best parameters: {'colsample_bytree': np.float64(0.9942601816442402), 'gamma': np.float64(0.1210276357557502), 'learning_rate': np.float64(0.21164066422176356), 'max_depth': 5, 'n_estimators': 283, 'subsample': np.float64(0.6950550175969599)} - Best embeddings: orginal - {'face_vertical_length_norm', 'flag_mouth', 'flag_eye1', 'face_angle_horizontal', 'flag_nose', 'y_eye2', 'x_left_elbow', 'x_left_hip', 'x_eye2', 'nose_hip_right', 'flag_eye2'} - Performance: 0.90 with all feature, 0.89 with 25_top_features, 0.84 with top_10_features

```
[3]: embeddings = emb_builder.create_embedding(flags=True,positions=True,␣
     ↪positions_normalized=True, geometric_info=True,k_positions_normalized=True␣
     ↪,k_geometric_info=True)
```

```
    Embedding
    creation-----------------------------------------------------------------
    Features: ['flag_eye1', 'flag_eye2', 'flag_nose', 'flag_mouth', 'x_eye1',
    'y_eye1', 'x_eye2', 'y_eye2', 'x_nose', 'y_nose', 'x_mouth', 'y_mouth',
```

```
'x_eye1_norm', 'y_eye1_norm', 'x_eye2_norm', 'y_eye2_norm', 'x_nose_norm',
'y_nose_norm', 'x_mouth_norm', 'y_mouth_norm', 'eye_distance',
'eye_distance_norm', 'face_vertical_length', 'face_vertical_length_norm',
'face_angle_vertical', 'face_angle_horizontal', 'symmetry_diff', 'head_ration',
'x_nose_k', 'y_nose_k', 'x_left_eye_k', 'y_left_eye_k', 'x_right_eye_k',
'y_right_eye_k', 'x_left_ear', 'y_left_ear', 'x_right_ear', 'y_right_ear',
'x_left_shoulder', 'y_left_shoulder', 'x_right_shoulder', 'y_right_shoulder',
'x_left_elbow', 'y_left_elbow', 'x_right_elbow', 'y_right_elbow',
'x_left_wrist', 'y_left_wrist', 'x_right_wrist', 'y_right_wrist', 'x_left_hip',
'y_left_hip', 'x_right_hip', 'y_right_hip', 'x_left_knee', 'y_left_knee',
'x_right_knee', 'y_right_knee', 'x_left_ankle', 'y_left_ankle', 'x_right_ankle',
'y_right_ankle', 'shoulders_dist', 'shoulder_hip_right_dist',
'shoulder_hip_left_dist', 'nose_shoulder_right', 'nose_shoulder_left',
'shoulder_left_knee_right', 'shoulder_right_knee_left', 'knee_ankle_right',
'knee_ankle_left', 'nose_hip_right', 'nose_hip_left',
'elbow_shoulder_hip_right', 'elbow_shoulder_hip_left',
'shoulder_elbow_wrist_right', 'shoulder_elbow_wrist_left',
'shoulder_hip_knee_right', 'shoulder_hip_knee_left', 'hip_knee_ankle_right',
'hip_knee_ankle_left', 'shoulders_line_inclination', 'hips_line_inclination',
'torsion']
FINISHED: 4158 embedding created
----------------------------------------------------------------------------
----------
```

```python
features_to_drop = ['face_vertical_length_norm', 'flag_mouth', 'flag_eye1',
 ↪'face_angle_horizontal', 'flag_nose', 'y_eye2', 'x_left_elbow',
 ↪'x_left_hip', 'x_eye2', 'nose_hip_right', 'flag_eye2']
embeddings = embeddings.drop(columns=features_to_drop)
clf = Classifier(embeddings, emb_builder.y, emb_builder.classes_bs,
 ↪image_paths=emb_builder.image_paths)

best_params = {
    'colsample_bytree': np.float64(0.9942601816442402),
    'gamma': np.float64(0.1210276357557502),
    'learning_rate': np.float64(0.21164066422176356),
    'max_depth': 5,
    'n_estimators': 283,
    'subsample': np.float64(0.6950550175969599),
    'use_label_encoder': False,  # necessario con sklearn>=1.0
    'eval_metric': 'logloss'
}
model = XGBClassifier(**best_params)

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    results =clf.evaluation_pipeline_save_misclassified(model)
```

```
----------
------------------------------------FIRST
ANALYSIS------------------------------------
--------------------------------------------------------------------------------
----------
```

/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of

glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28 after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >= 2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain features such as GPU algorithms or federated learning are not available. To use these features, please upgrade to a recent Linux distro with glibc 2.28+, and install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-packages/xgboost/core.py:377: FutureWarning: Your system has an old version of glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28 after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >= 2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain features such as GPU algorithms or federated learning are not available. To use these features, please upgrade to a recent Linux distro with glibc 2.28+, and install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-packages/xgboost/core.py:377: FutureWarning: Your system has an old version of glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28 after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >= 2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain features such as GPU algorithms or federated learning are not available. To use these features, please upgrade to a recent Linux distro with glibc 2.28+, and install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-packages/xgboost/core.py:377: FutureWarning: Your system has an old version of glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28 after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >= 2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain features such as GPU algorithms or federated learning are not available. To use these features, please upgrade to a recent Linux distro with glibc 2.28+, and install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-packages/xgboost/core.py:377: FutureWarning: Your system has an old version of glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28 after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >= 2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain features such as GPU algorithms or federated learning are not available. To use these features, please upgrade to a recent Linux distro with glibc 2.28+, and install the 'manylinux_2_28' variant.
  warnings.warn(

```
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
/home/terra/anaconda3/envs/SIDS_project/lib/python3.10/site-
packages/xgboost/core.py:377: FutureWarning: Your system has an old version of
glibc (< 2.28). We will stop supporting Linux distros with glibc older than 2.28
after **May 31, 2025**. Please upgrade to a recent Linux distro (with glibc >=
2.28) to use future versions of XGBoost.
Note: You have installed the 'manylinux2014' variant of XGBoost. Certain
features such as GPU algorithms or federated learning are not available. To use
these features, please upgrade to a recent Linux distro with glibc 2.28+, and
install the 'manylinux_2_28' variant.
  warnings.warn(
```
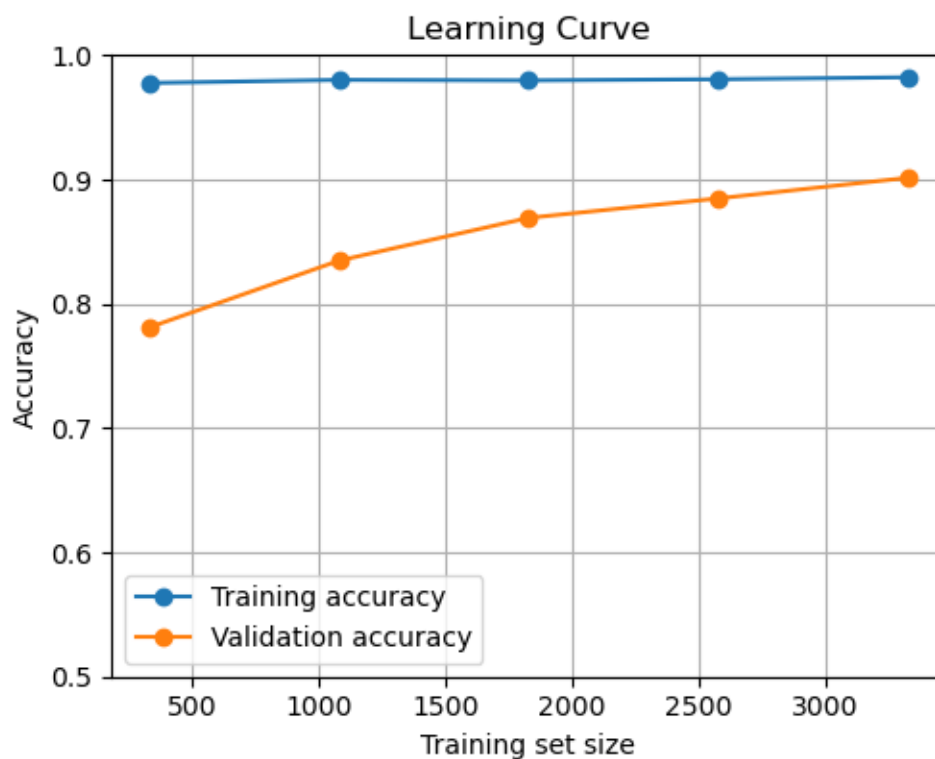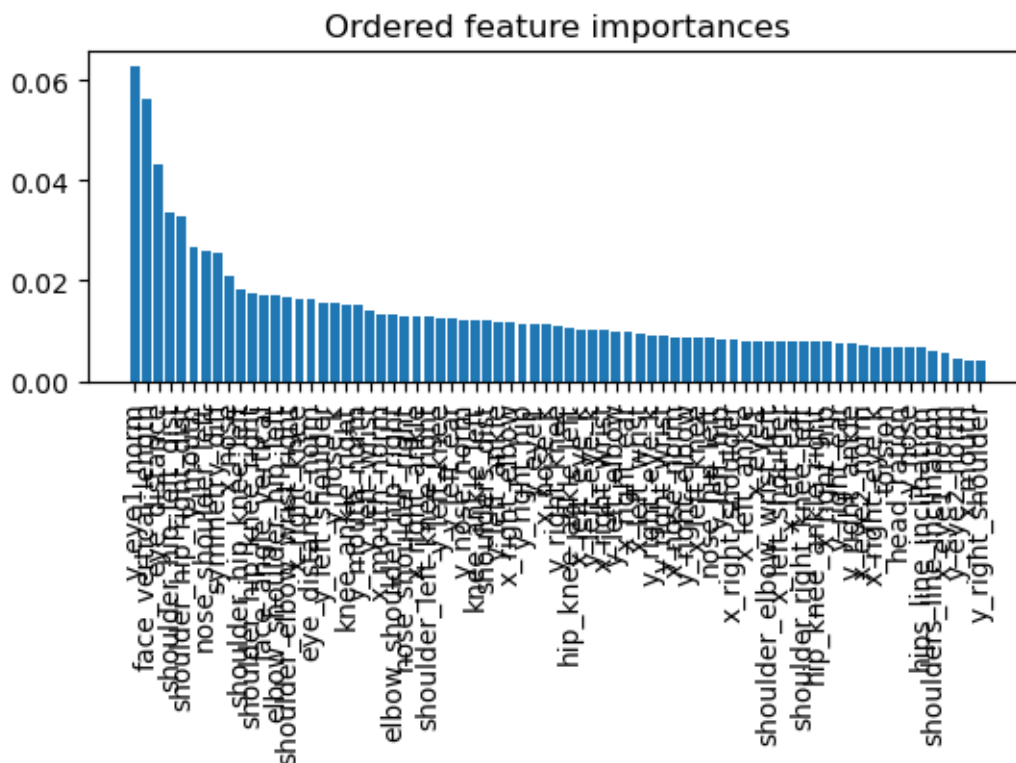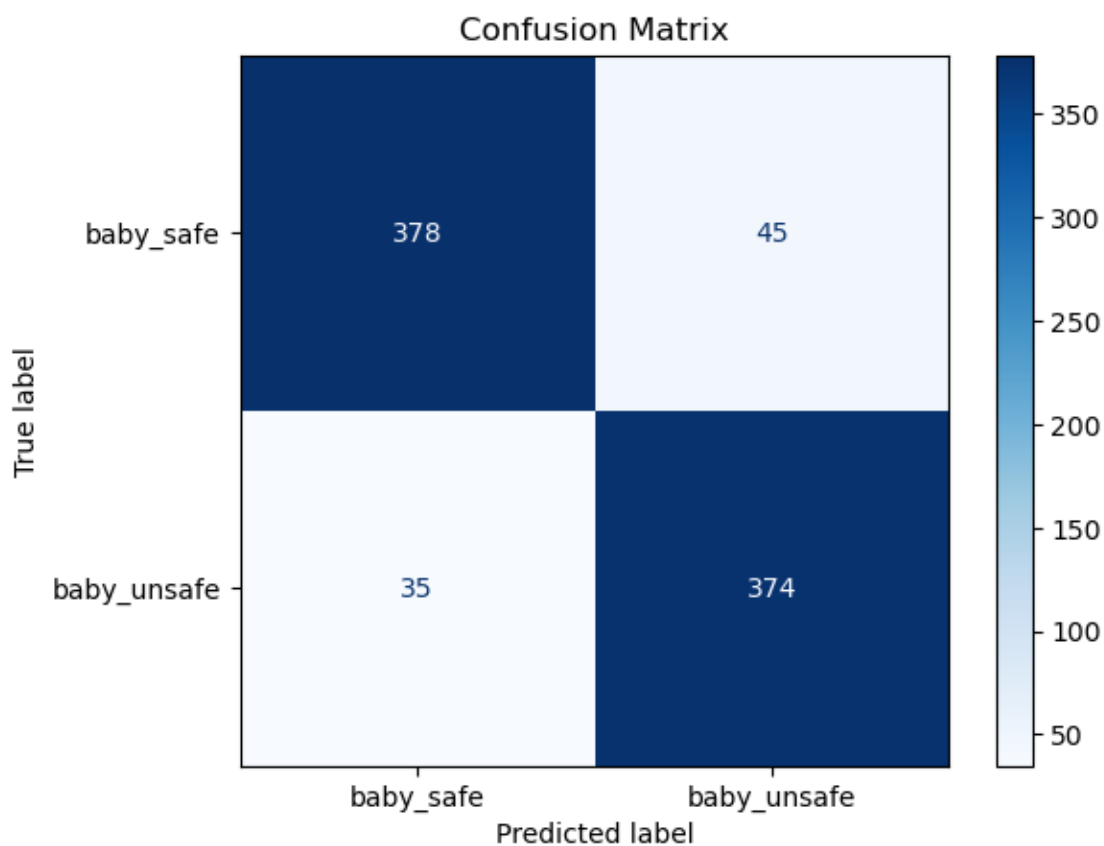
Ordered feature importances

Learning Curve

```
Dataset labels:-------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report----------------------------------------------
              precision    recall  f1-score   support

   baby_safe       0.92      0.89      0.90       423
 baby_unsafe       0.89      0.91      0.90       409

    accuracy                           0.90       832
   macro avg       0.90      0.90      0.90       832
weighted avg       0.90      0.90      0.90       832

Confusion matrix-------------------------------------
```
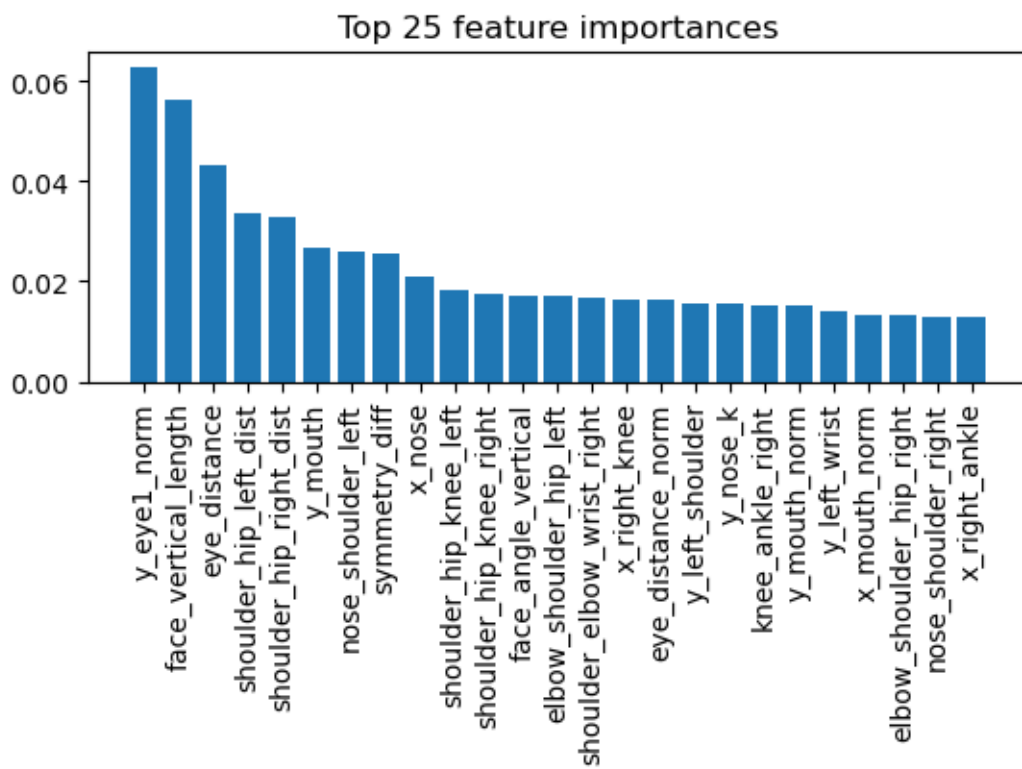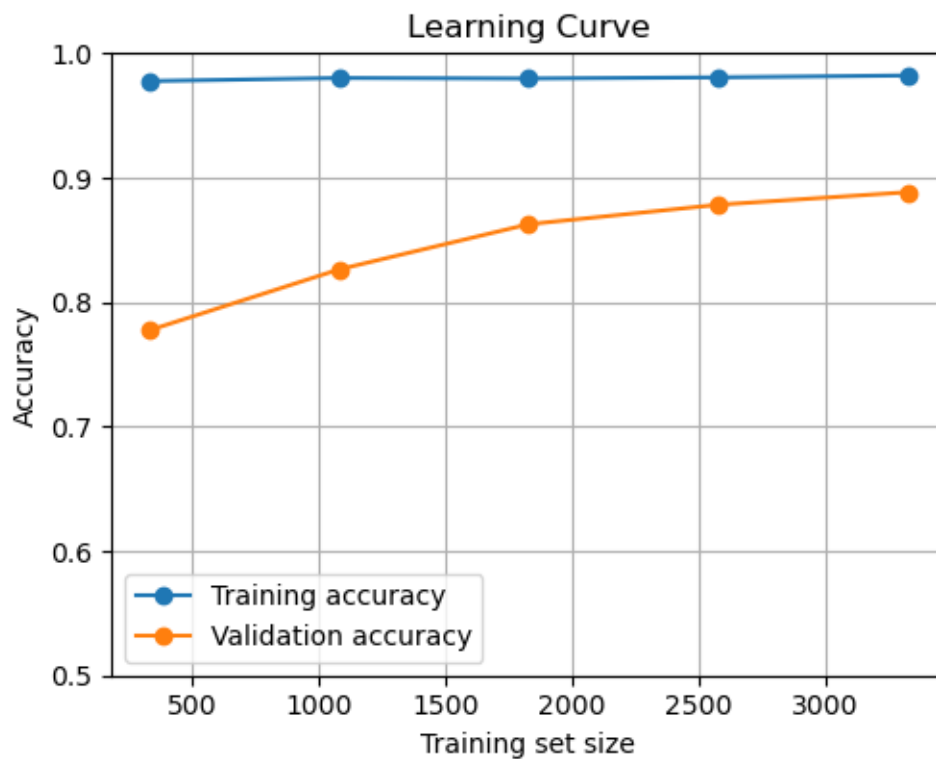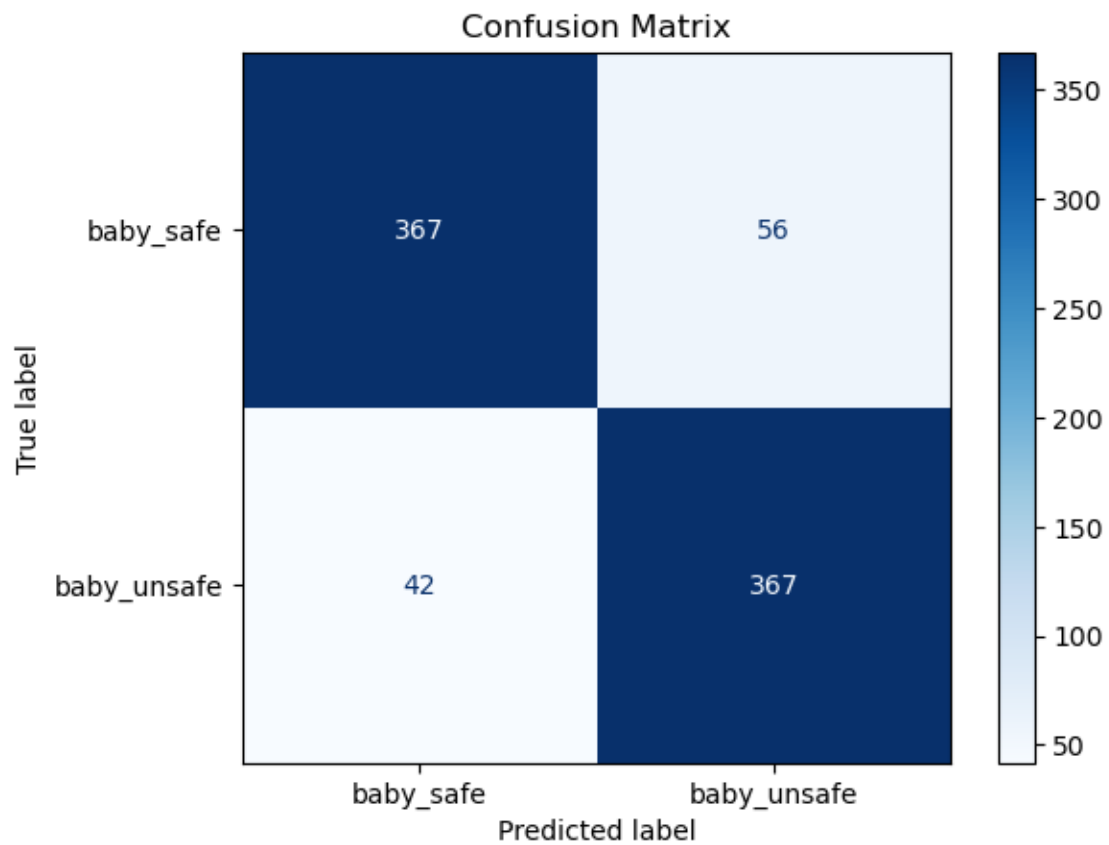

Confusion Matrix

Top 25 feature importances

Learning Curve

```
Dataset labels:-------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report----------------------------------------------
              precision    recall   f1-score   support

   baby_safe       0.90      0.87       0.88       423
 baby_unsafe       0.87      0.90       0.88       409

    accuracy                            0.88       832
   macro avg       0.88      0.88       0.88       832
weighted avg       0.88      0.88       0.88       832

Confusion matrix------------------------------------
```
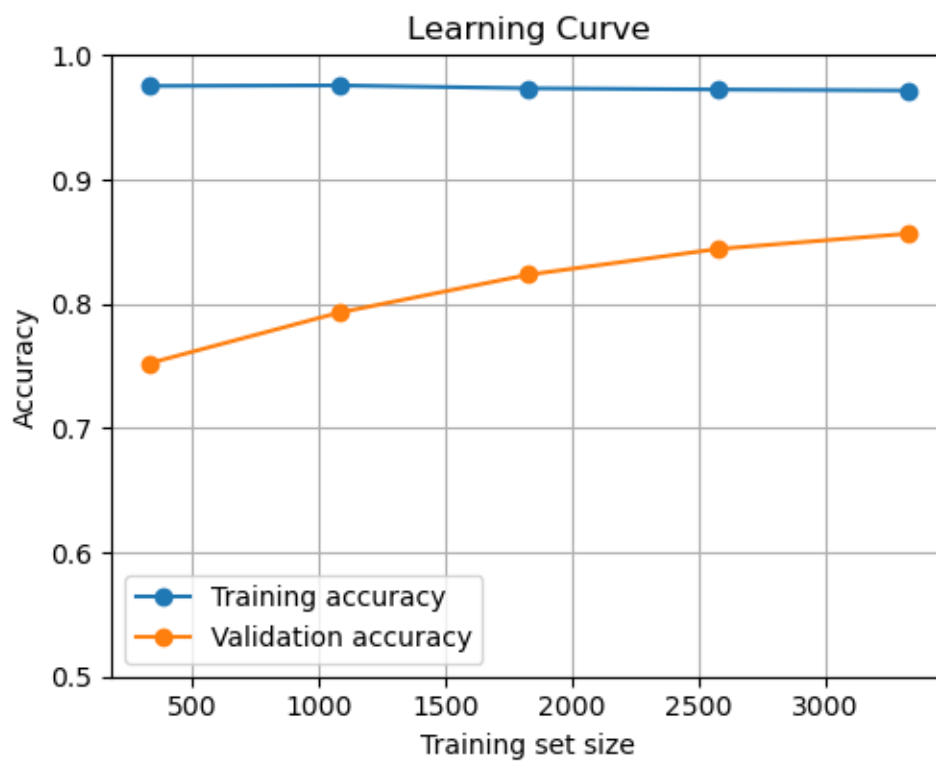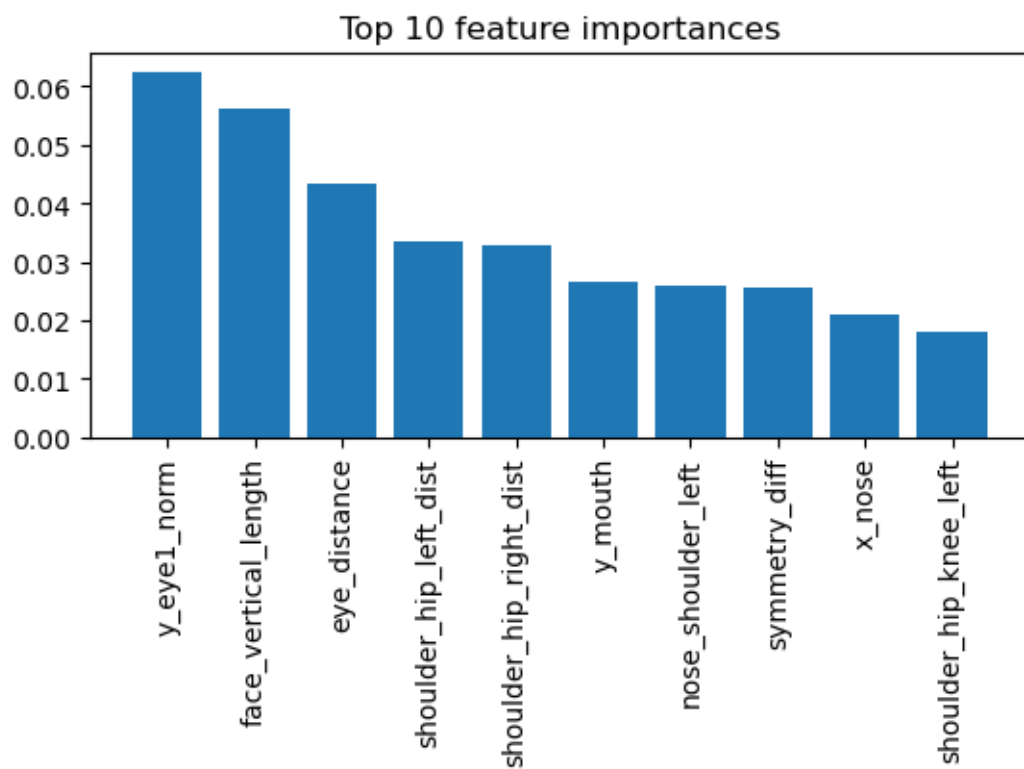
Confusion Matrix

-------------------------------TOP 10 FEATURES
ANALYSIS--------------------------------

Top 10 feature importances

Learning Curve

```
Dataset labels:-------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report---------------------------------------------
              precision    recall  f1-score   support

   baby_safe       0.85      0.83      0.84       423
 baby_unsafe       0.83      0.85      0.84       409

    accuracy                          0.84       832
   macro avg       0.84      0.84      0.84       832
weighted avg       0.84      0.84      0.84       832

Confusion matrix------------------------------------
```
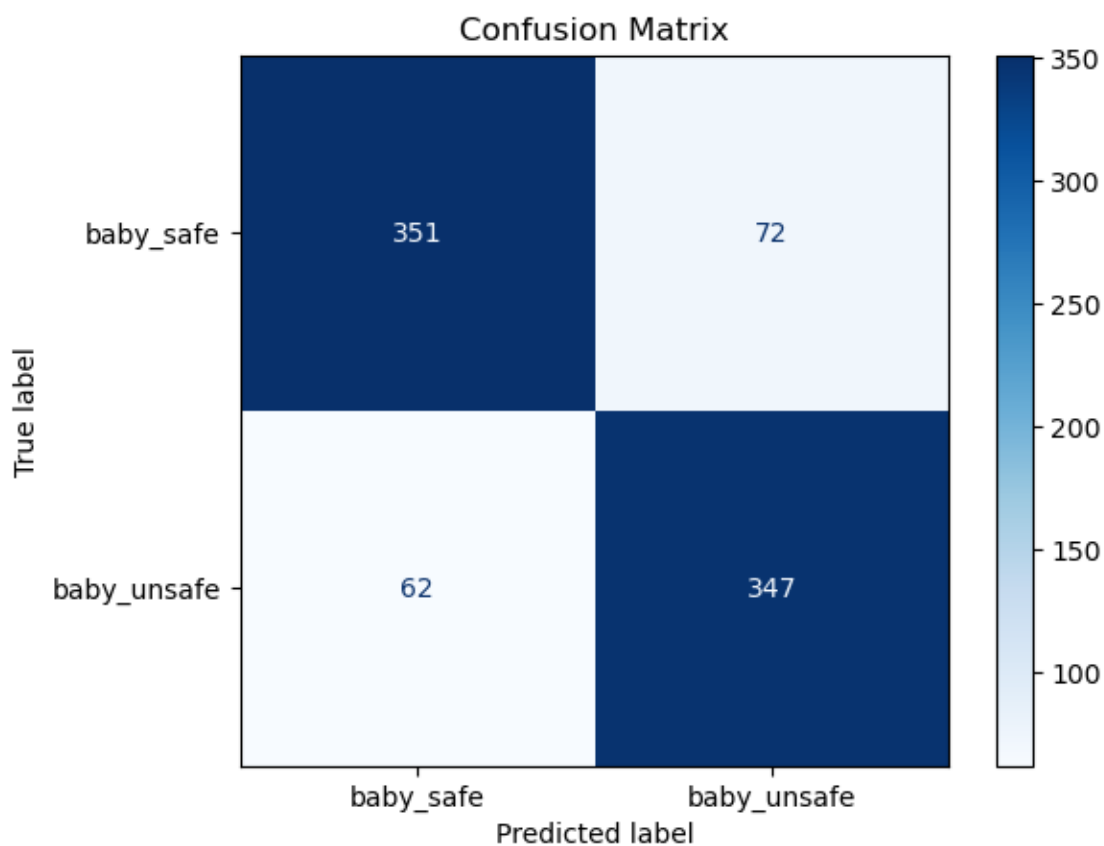

Confusion Matrix

```python
[5]: prediction = results["all_features"]["y_predicted"]
     true_y= clf.y_test
     misclassified = np.where(true_y != prediction)[0]

     misclassified_images = [clf.images_paths_test[i] for i in misclassified]
     image_dataset_path=emb_builder.dataset

     folder_path = f"{project_dir}image_prediction/approach_features_selection/"
     if not os.path.exists(folder_path):
         os.makedirs(folder_path)

     from PIL import Image, ImageDraw, ImageFont

     for img_path, prediction in zip(misclassified_images, prediction):
             img = Image.open(f"{image_dataset_path}/{img_path}")
             draw = ImageDraw.Draw(img)

             try:
                 font = ImageFont.truetype("DejaVuSans-Bold.ttf", size=34)   #␣
     ↪Imposta la dimensione del font
             except IOError:
                 font = ImageFont.load_default()  # Usa il font di default se il␣
     ↪file ttf non è trovato

             predicted_class = [key for key, value in emb_builder.classes_bs.items()␣
     ↪if value == prediction][0]
             text = f"{predicted_class}"
             text_position = (50, 50)
             text_color = (255, 0, 0)
             draw.text(text_position, text, fill=text_color, font=font)

             new_image_path = f"{folder_path}{img_path}"
             img.save(new_image_path)

     print(f"Misclassified images successfully saved in {folder_path}")
```

Misclassified images successfully saved in /home/terra/Desktop/unimore/AI_engine
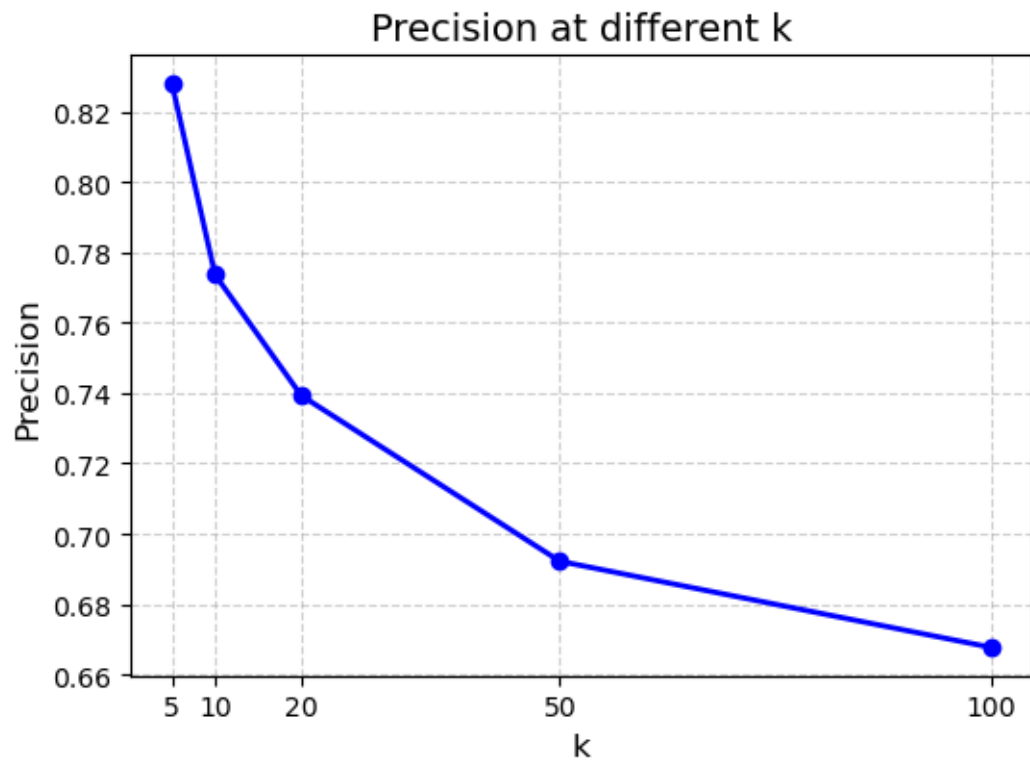ering/SIDS_revelation_project/image_prediction/approach_features_selection/

```python
[6]: ret = ImageRetrieval(embeddings, emb_builder.y, emb_builder.image_paths,␣
     ↪emb_builder.dataset, emb_builder.classes_bs)
     ret.report('euclidean')
```

Precision at different
k:--

Precision at different k

Recall at
R--
0.5437276847656125

Silhouette
score---

Distribuzione Silhouette score per classe
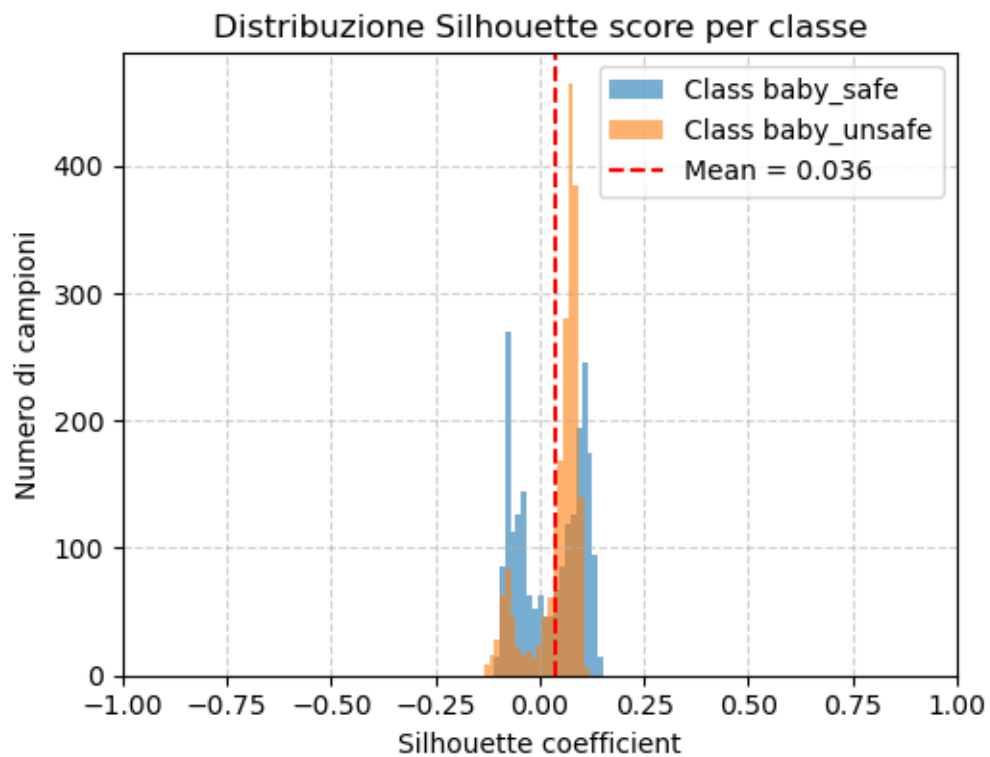
Silhouette score (euclidean): 0.036

Embeddings
distributions--
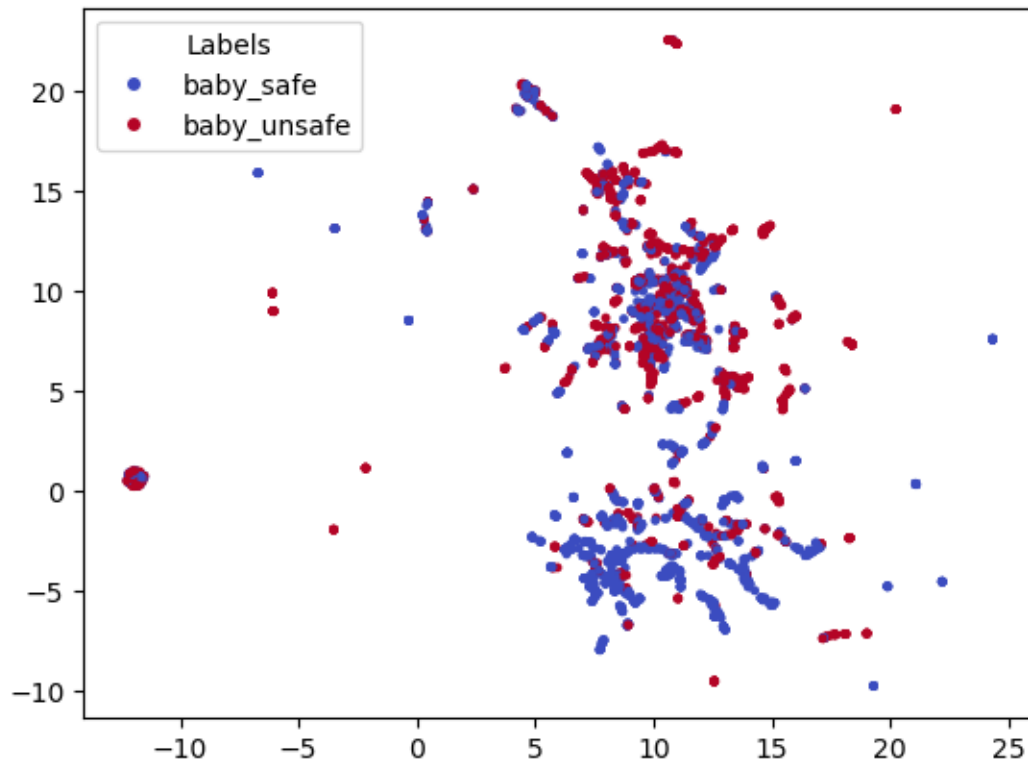
```
[7]:  image_paths = emb_builder.image_paths
      idx_query = 99
      image_to_retrieve = f"{emb_builder.dataset}/{image_paths[idx_query]}"

      print("Image to retrieve")
      img = mpimg.imread(image_to_retrieve)
      plt.figure(figsize=(3, 3))
      plt.imshow(img)
      plt.axis('off')
      plt.show()

      distances_all, image_paths_similar_all = ret.
        ↪retrieve_similar(idx_query=idx_query,k=5,verbose=False)
      ret.show_images(image_paths_similar_all)
```

Image to retrieve

Image 1   Image 2   Image 3   Image 4   Image 5

0.2   Best model with approach Supervised Learning Metric

XGBC with no optimization

```
[8]: embeddings = emb_builder.create_embedding(flags=True,positions=True,␣
     ↪positions_normalized=True, geometric_info=True,k_positions_normalized=True␣
     ↪,k_geometric_info=True)
```

```
Embedding
creation-------------------------------------------------------------------------
Features: ['flag_eye1', 'flag_eye2', 'flag_nose', 'flag_mouth', 'x_eye1',
'y_eye1', 'x_eye2', 'y_eye2', 'x_nose', 'y_nose', 'x_mouth', 'y_mouth',
'x_eye1_norm', 'y_eye1_norm', 'x_eye2_norm', 'y_eye2_norm', 'x_nose_norm',
'y_nose_norm', 'x_mouth_norm', 'y_mouth_norm', 'eye_distance',
'eye_distance_norm', 'face_vertical_length', 'face_vertical_length_norm',
'face_angle_vertical', 'face_angle_horizontal', 'symmetry_diff', 'head_ration',
'x_nose_k', 'y_nose_k', 'x_left_eye_k', 'y_left_eye_k', 'x_right_eye_k',
'y_right_eye_k', 'x_left_ear', 'y_left_ear', 'x_right_ear', 'y_right_ear',
'x_left_shoulder', 'y_left_shoulder', 'x_right_shoulder', 'y_right_shoulder',
'x_left_elbow', 'y_left_elbow', 'x_right_elbow', 'y_right_elbow',
```

18

```
'x_left_wrist', 'y_left_wrist', 'x_right_wrist', 'y_right_wrist', 'x_left_hip',
'y_left_hip', 'x_right_hip', 'y_right_hip', 'x_left_knee', 'y_left_knee',
'x_right_knee', 'y_right_knee', 'x_left_ankle', 'y_left_ankle', 'x_right_ankle',
'y_right_ankle', 'shoulders_dist', 'shoulder_hip_right_dist',
'shoulder_hip_left_dist', 'nose_shoulder_right', 'nose_shoulder_left',
'shoulder_left_knee_right', 'shoulder_right_knee_left', 'knee_ankle_right',
'knee_ankle_left', 'nose_hip_right', 'nose_hip_left',
'elbow_shoulder_hip_right', 'elbow_shoulder_hip_left',
'shoulder_elbow_wrist_right', 'shoulder_elbow_wrist_left',
'shoulder_hip_knee_right', 'shoulder_hip_knee_left', 'hip_knee_ankle_right',
'hip_knee_ankle_left', 'shoulders_line_inclination', 'hips_line_inclination',
'torsion']
FINISHED: 4158 embedding created
-----------------------------------------------------------------------------
----------
```
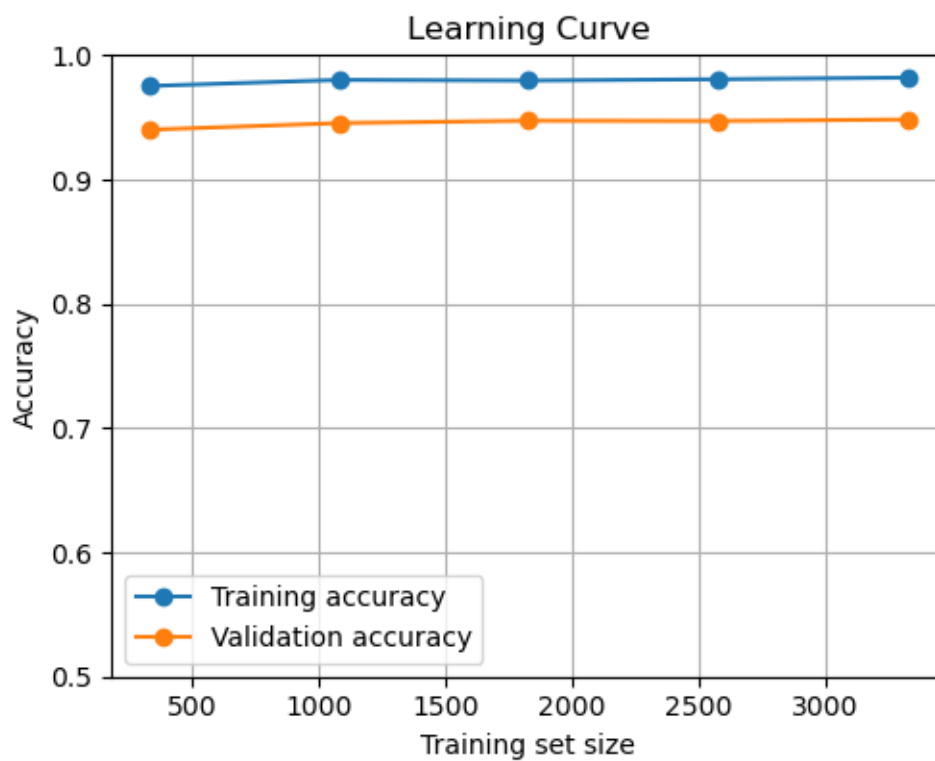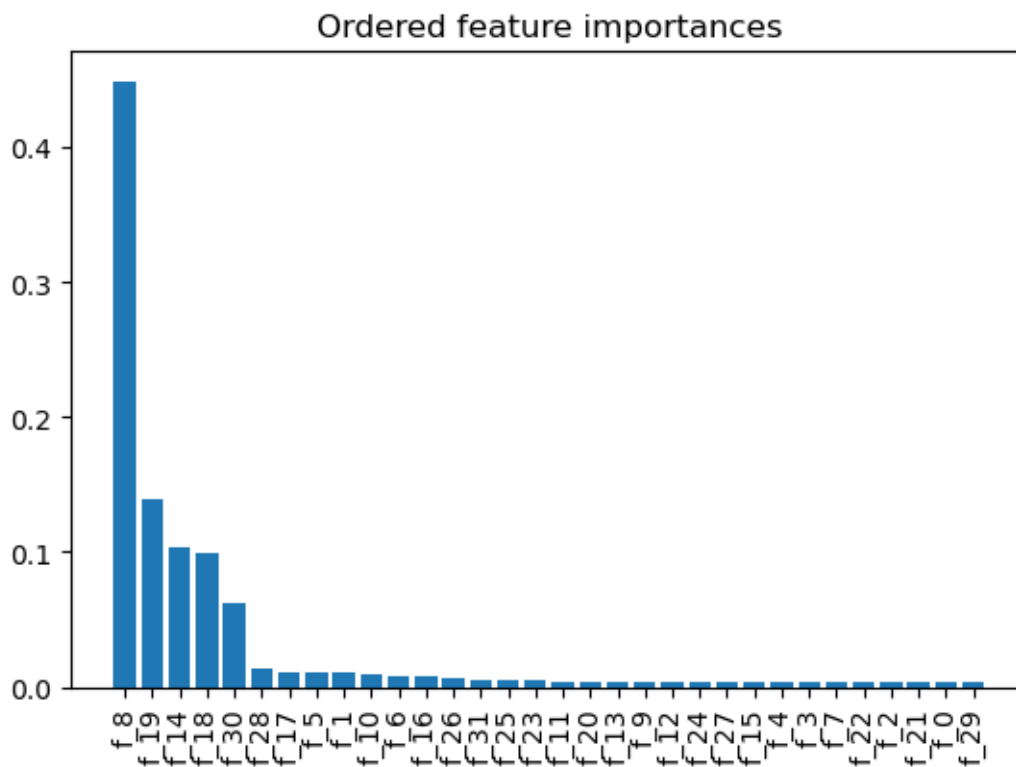
```python
[9]: dataset = EmbeddingDataset(embeddings.to_numpy(),emb_builder.y,device=device)
     model = dataset.train_embeddings(embed_dim=32, epochs=50, batch_size=128,␣
       ↪lr=1e-3,verbose=False,weight_decay=1e-7,dropout_rate=0.05)

     embeddings_new = dataset.extract_embeddings(model)
     embeddings_new= pd.DataFrame(embeddings_new.to_numpy(), columns=[f"f_{i}" for i␣
       ↪in range(embeddings_new.shape[1])])
     clf = Classifier(embeddings_new, emb_builder.y, emb_builder.classes_bs,␣
       ↪image_paths=emb_builder.image_paths)
```

```python
[10]: params = {
          'n_estimators': 300,
          'max_depth': 5,
          'learning_rate': 0.05,
          'subsample': 0.8,
          'colsample_bytree': 0.8,
          'reg_lambda': 1,
          'reg_alpha': 0.5,
          'random_state': None
      }
      model = XGBClassifier(**params)

      with warnings.catch_warnings():
          warnings.simplefilter("ignore")
          results =clf.evaluation_pipeline_save_misclassified(model)
```

```
-----------------------------------------------------------------------------
----------
------------------------------------FIRST
ANALYSIS-------------------------------------
-----------------------------------------------------------------------------
----------
```

Ordered feature importances

Learning Curve

```
Dataset labels:--------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report-----------------------------------------------
              precision    recall  f1-score    support

   baby_safe       0.96      0.94      0.95        423
 baby_unsafe       0.94      0.96      0.95        409

    accuracy                          0.95        832
   macro avg       0.95      0.95      0.95        832
weighted avg       0.95      0.95      0.95        832

Confusion matrix-------------------------------------
```
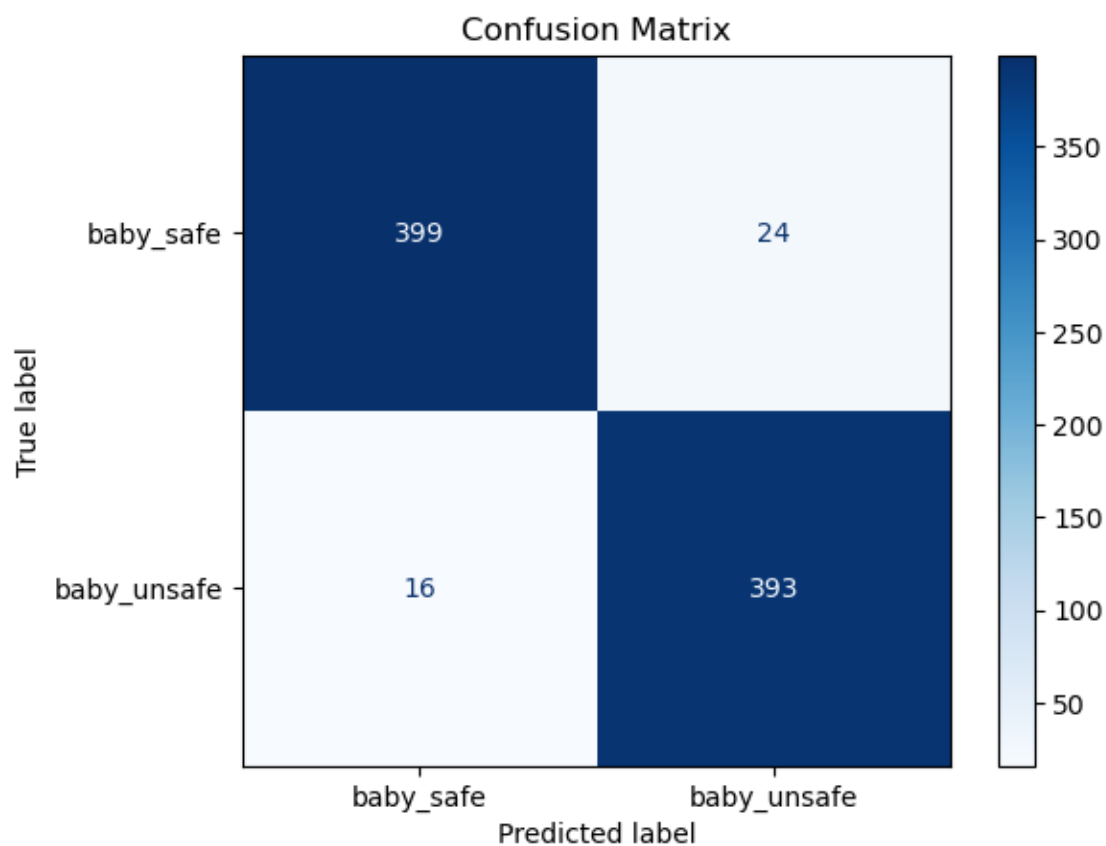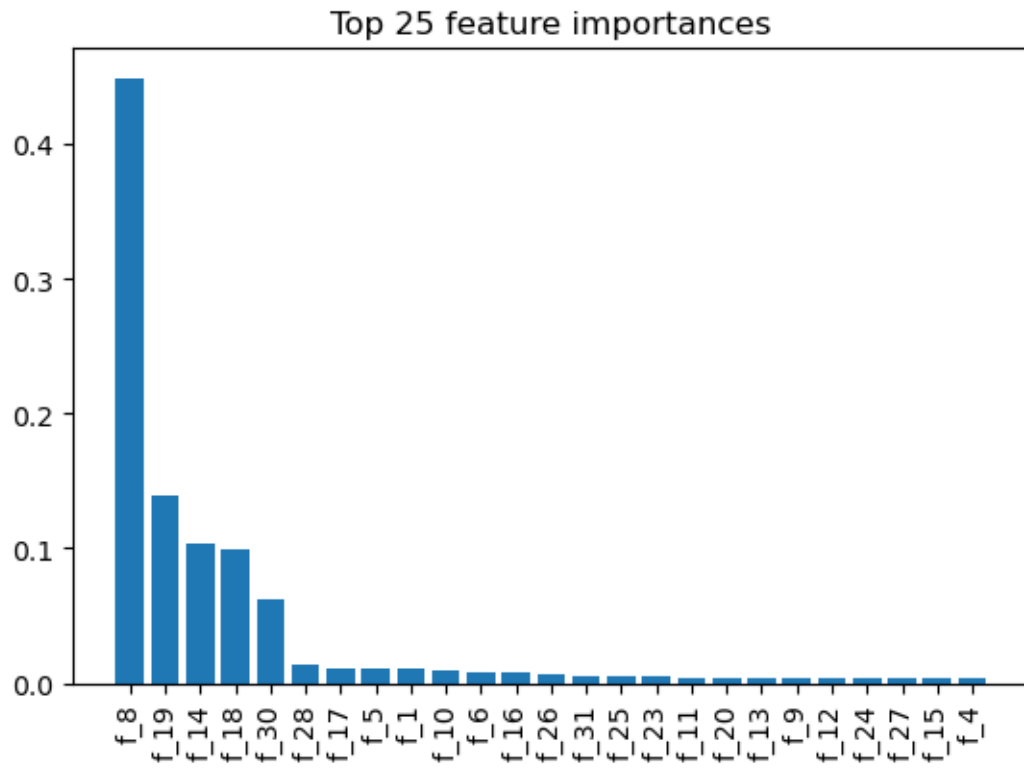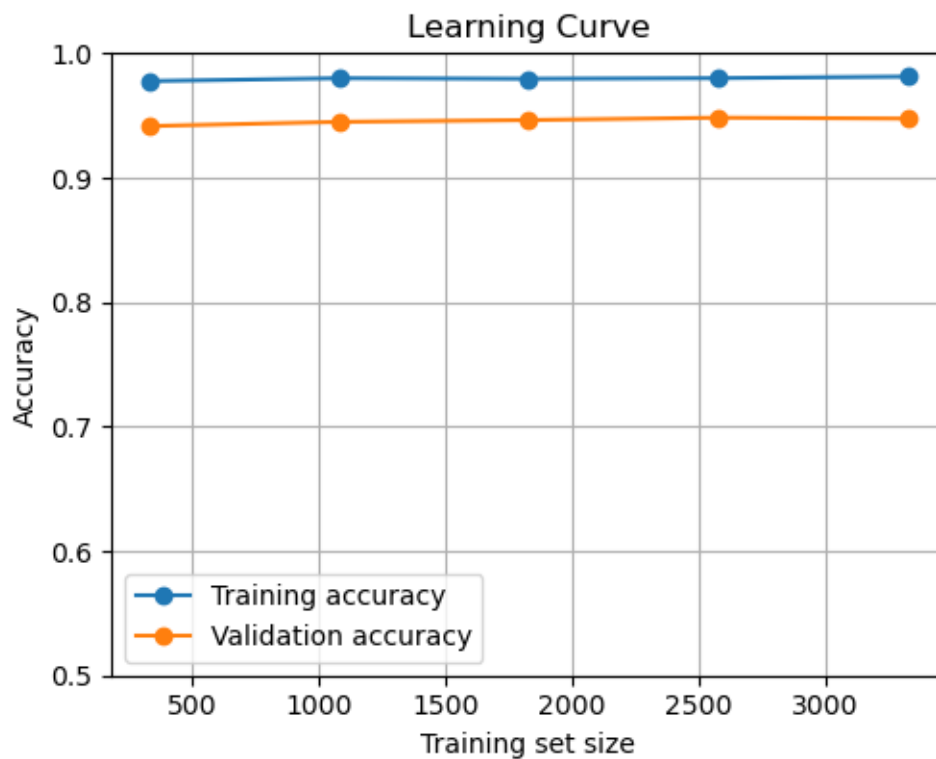

Confusion Matrix

--------------------------------TOP 25 FEATURES
ANALYSIS--------------------------------

Top 25 feature importances

Learning Curve

```
Dataset labels:-------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report---------------------------------------------
              precision    recall  f1-score   support

   baby_safe       0.96      0.94      0.95       423
 baby_unsafe       0.94      0.96      0.95       409

    accuracy                          0.95       832
   macro avg       0.95      0.95      0.95       832
weighted avg       0.95      0.95      0.95       832

Confusion matrix-----------------------------------
```
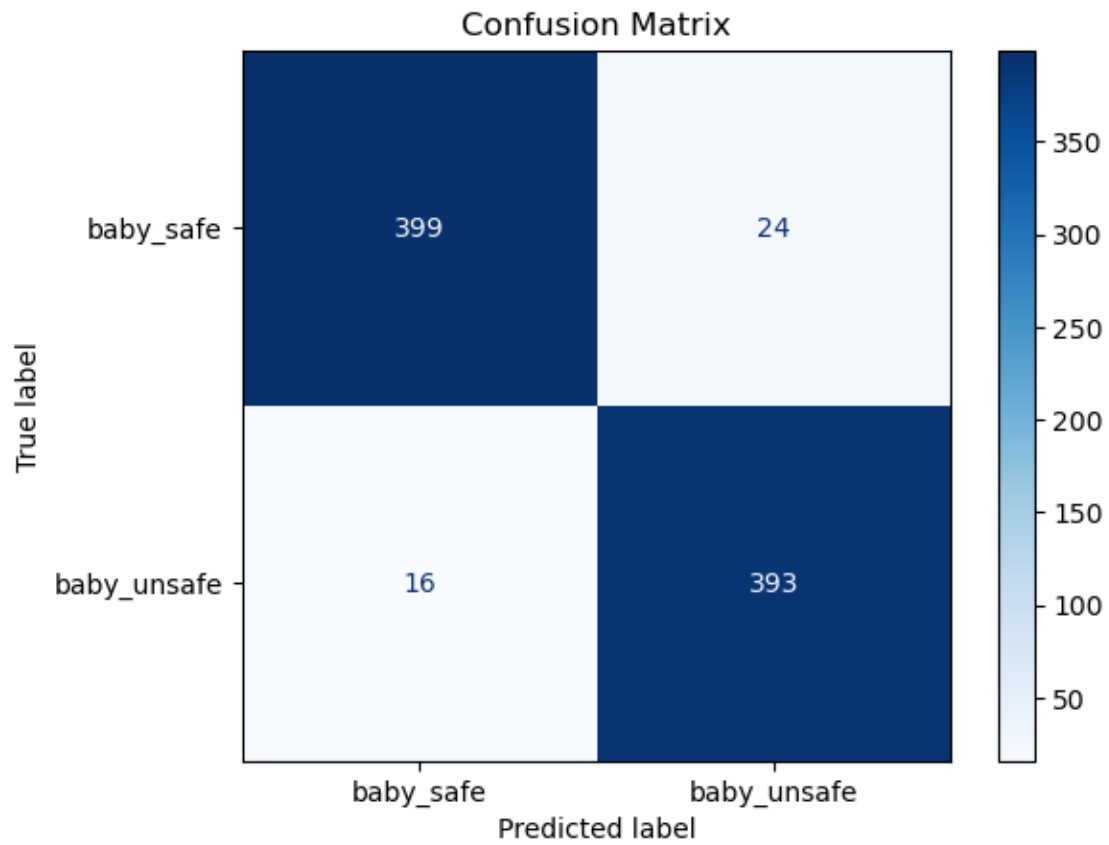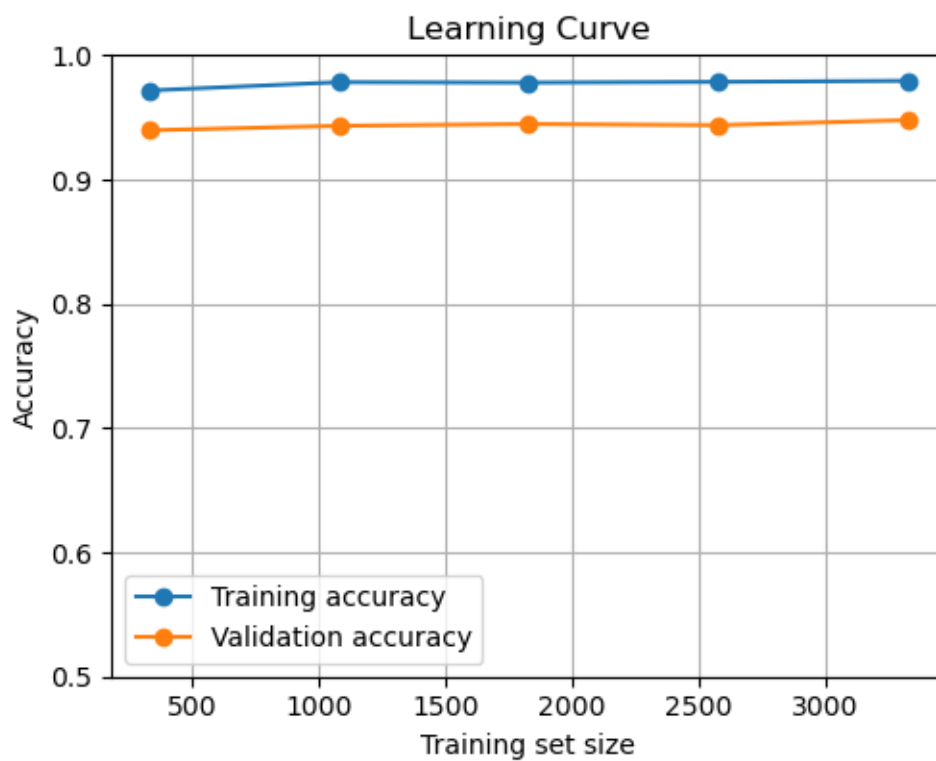
Confusion Matrix

--------------------------------TOP 10 FEATURES
ANALYSIS--------------------------------

Top 10 feature importances

Learning Curve

```
Dataset labels:--------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report----------------------------------------------
              precision    recall  f1-score   support

   baby_safe       0.96      0.93      0.95       423
 baby_unsafe       0.93      0.96      0.95       409

    accuracy                          0.95       832
   macro avg       0.95      0.95      0.95       832
weighted avg       0.95      0.95      0.95       832


Confusion matrix------------------------------------
```
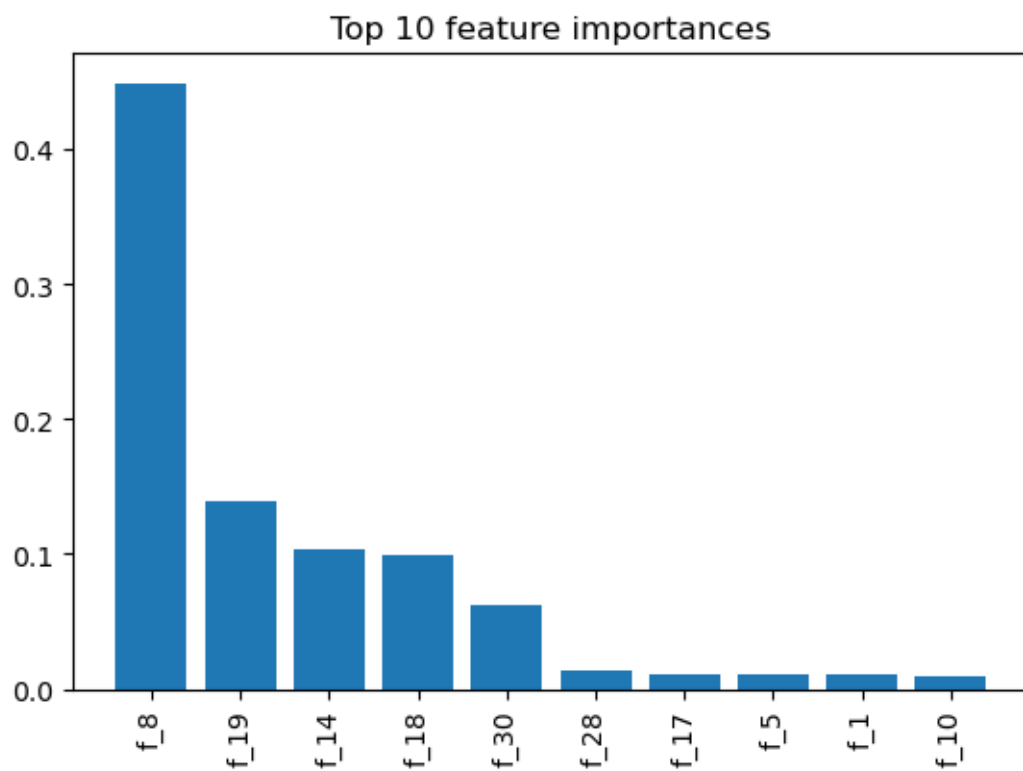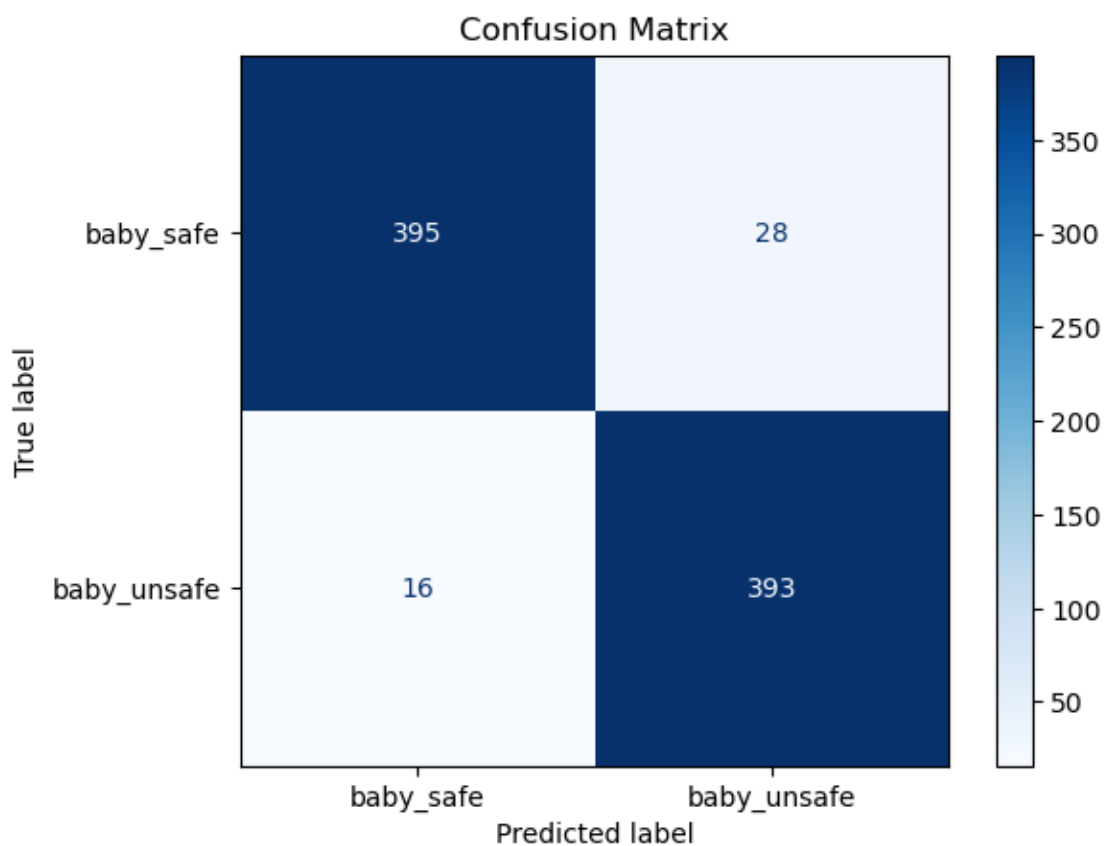
Confusion Matrix

	Predicted baby_safe	Predicted baby_unsafe
True baby_safe	395	28
True baby_unsafe	16	393

True label / Predicted label

```python
[11]: prediction = results["all_features"]["y_predicted"]
      true_y= clf.y_test
      misclassified = np.where(true_y != prediction)[0]

      misclassified_images = [clf.images_paths_test[i] for i in misclassified]
      image_dataset_path=emb_builder.dataset

      folder_path = f"{project_dir}image_prediction/
       ↪approach_supervised_learning_metric/"
      if not os.path.exists(folder_path):
          os.makedirs(folder_path)

      from PIL import Image, ImageDraw, ImageFont

      for img_path, prediction in zip(misclassified_images, prediction):
              img = Image.open(f"{image_dataset_path}/{img_path}")
              draw = ImageDraw.Draw(img)

              try:
                  font = ImageFont.truetype("DejaVuSans-Bold.ttf", size=34)  #␣
       ↪Imposta la dimensione del font
              except IOError:
                  font = ImageFont.load_default()  # Usa il font di default se il␣
       ↪file ttf non è trovato

              predicted_class = [key for key, value in emb_builder.classes_bs.items()␣
       ↪if value == prediction][0]
              text = f"{predicted_class}"
              text_position = (50, 50)
              text_color = (255, 0, 0)
              draw.text(text_position, text, fill=text_color, font=font)

              new_image_path = f"{folder_path}{img_path}"
              img.save(new_image_path)

      print(f"Misclassified images successfully saved in {folder_path}")
```

Misclassified images successfully saved in /home/terra/Desktop/unimore/AI_engine
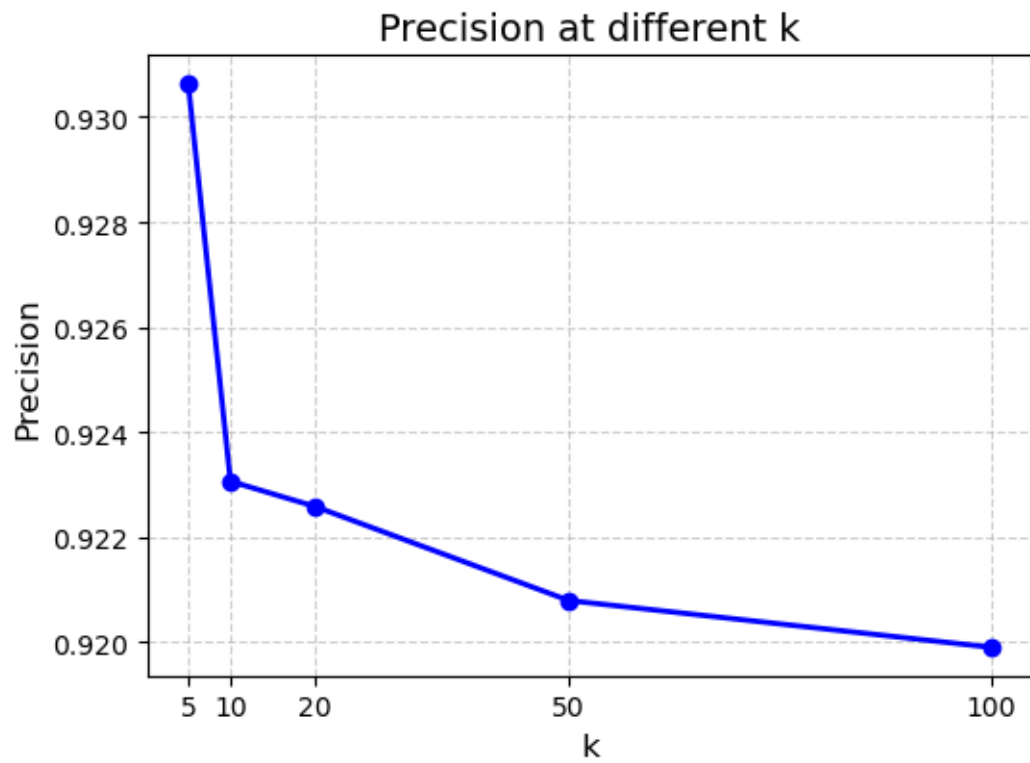ering/SIDS_revelation_project/image_prediction/approach_supervised_learning_metr
ic/

```python
[12]: ret = ImageRetrieval(embeddings_new, emb_builder.y, emb_builder.image_paths,␣
       ↪emb_builder.dataset, emb_builder.classes_bs)
      ret.report('euclidean')
```
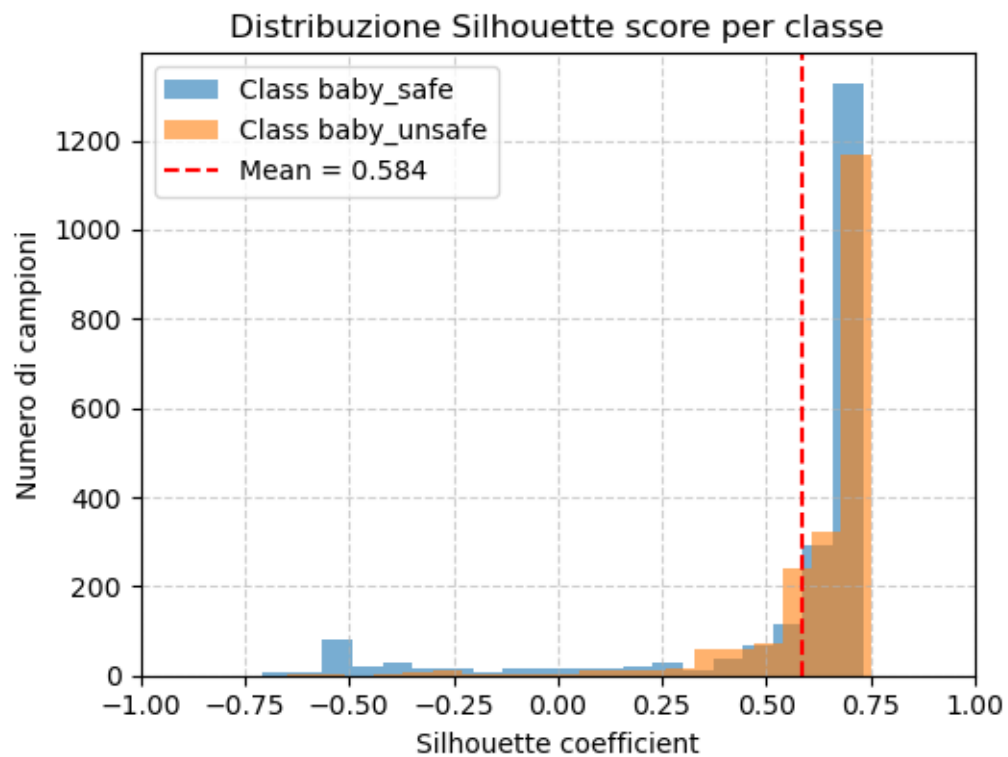
Precision at different
k:--

Precision at different k

Recall at
R--
0.8750370384654548

Silhouette
score--

Distribuzione Silhouette score per classe

Silhouette score (euclidean): 0.584

Embeddings
distributions--
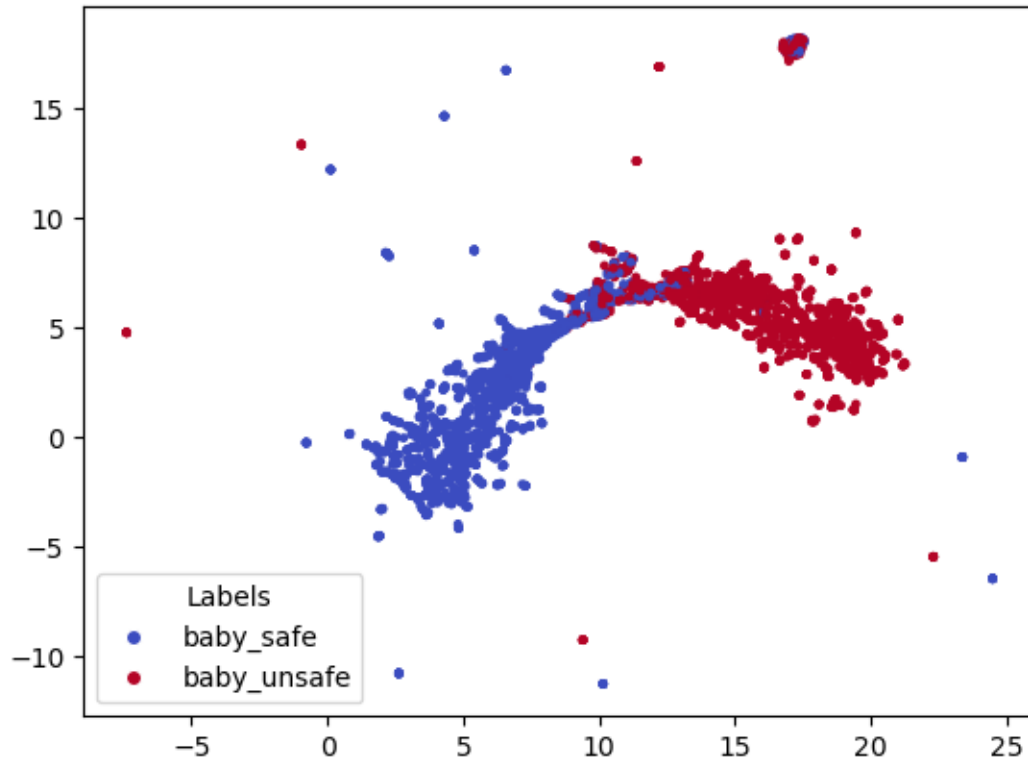
```
[13]:  image_paths = emb_builder.image_paths
       idx_query = 99
       image_to_retrieve = f"{emb_builder.dataset}/{image_paths[idx_query]}"

       print("Image to retrieve")
       img = mpimg.imread(image_to_retrieve)
       plt.figure(figsize=(3, 3))
       plt.imshow(img)
       plt.axis('off')
       plt.show()

       distances_all, image_paths_similar_all = ret.
         ↪retrieve_similar(idx_query=idx_query,k=5,verbose=False)
       ret.show_images(image_paths_similar_all)
```

Image to retrieve

0.3 Best model with approach Supervised Learning Metric

XGBC with optimization

Best parameters : {'colsample_bytree': np.float64(0.6547542520275229), 'gamma': np.float64(0.45000932092405255), 'learning_rate': np.float64(0.2721670232687546), 'max_depth': 4, 'n_estimators': 367, 'subsample': np.float64(0.853239225342979)} Best mean cross-validation accuracy: 0.9735415114362482

```
[14]: embeddings = emb_builder.create_embedding(flags=True,positions=True,
      ↪positions_normalized=True, geometric_info=True,k_positions_normalized=True
      ↪,k_geometric_info=True)
```

```
Embedding
creation----------------------------------------------------------------------
Features: ['flag_eye1', 'flag_eye2', 'flag_nose', 'flag_mouth', 'x_eye1',
'y_eye1', 'x_eye2', 'y_eye2', 'x_nose', 'y_nose', 'x_mouth', 'y_mouth',
'x_eye1_norm', 'y_eye1_norm', 'x_eye2_norm', 'y_eye2_norm', 'x_nose_norm',
'y_nose_norm', 'x_mouth_norm', 'y_mouth_norm', 'eye_distance',
'eye_distance_norm', 'face_vertical_length', 'face_vertical_length_norm',
'face_angle_vertical', 'face_angle_horizontal', 'symmetry_diff', 'head_ration',
```

```
'x_nose_k', 'y_nose_k', 'x_left_eye_k', 'y_left_eye_k', 'x_right_eye_k',
'y_right_eye_k', 'x_left_ear', 'y_left_ear', 'x_right_ear', 'y_right_ear',
'x_left_shoulder', 'y_left_shoulder', 'x_right_shoulder', 'y_right_shoulder',
'x_left_elbow', 'y_left_elbow', 'x_right_elbow', 'y_right_elbow',
'x_left_wrist', 'y_left_wrist', 'x_right_wrist', 'y_right_wrist', 'x_left_hip',
'y_left_hip', 'x_right_hip', 'y_right_hip', 'x_left_knee', 'y_left_knee',
'x_right_knee', 'y_right_knee', 'x_left_ankle', 'y_left_ankle', 'x_right_ankle',
'y_right_ankle', 'shoulders_dist', 'shoulder_hip_right_dist',
'shoulder_hip_left_dist', 'nose_shoulder_right', 'nose_shoulder_left',
'shoulder_left_knee_right', 'shoulder_right_knee_left', 'knee_ankle_right',
'knee_ankle_left', 'nose_hip_right', 'nose_hip_left',
'elbow_shoulder_hip_right', 'elbow_shoulder_hip_left',
'shoulder_elbow_wrist_right', 'shoulder_elbow_wrist_left',
'shoulder_hip_knee_right', 'shoulder_hip_knee_left', 'hip_knee_ankle_right',
'hip_knee_ankle_left', 'shoulders_line_inclination', 'hips_line_inclination',
'torsion']
FINISHED: 4158 embedding created
--------------------------------------------------------------------------------
----------
```

```python
dataset = EmbeddingDataset(embeddings.to_numpy(),emb_builder.y,device=device)
model = dataset.train_embeddings(embed_dim=32, epochs=50, batch_size=128,
 ↪lr=1e-3,verbose=False,weight_decay=1e-7,dropout_rate=0.05)

embeddings_new = dataset.extract_embeddings(model)
embeddings_new= pd.DataFrame(embeddings_new.to_numpy(), columns=[f"f_{i}" for i
 ↪in range(embeddings_new.shape[1])])
clf = Classifier(embeddings_new, emb_builder.y, emb_builder.classes_bs,
 ↪image_paths=emb_builder.image_paths)
```
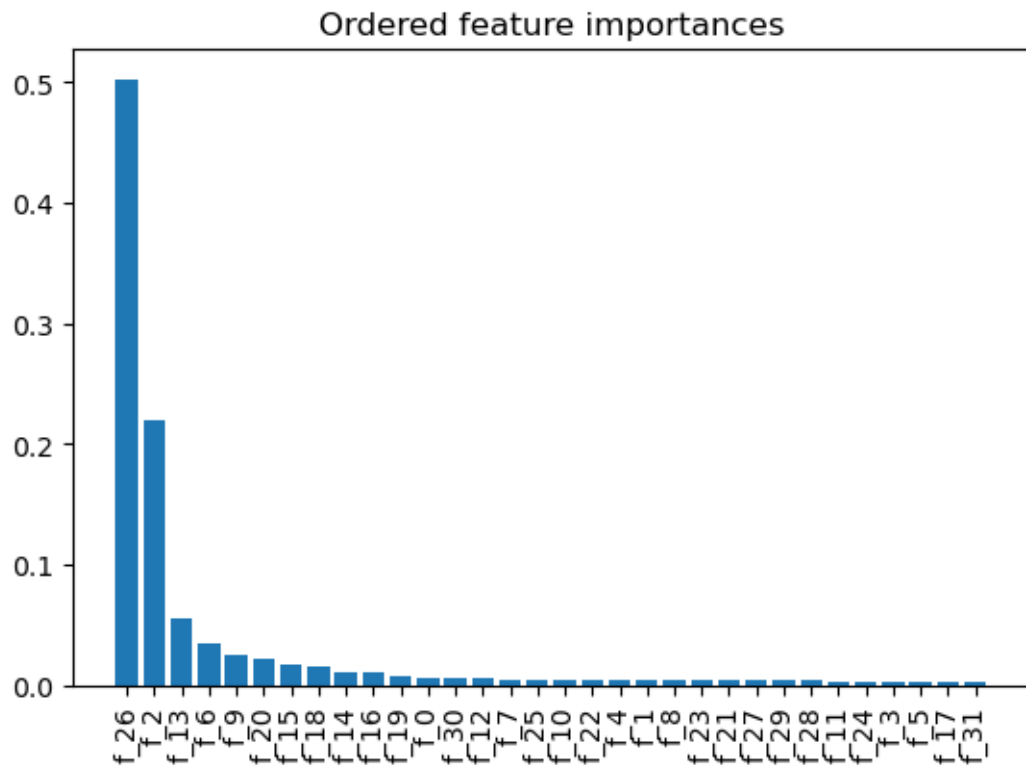
```python
best_params =  {
    'colsample_bytree': np.float64(0.6547542520275229),
    'gamma': np.float64(0.45000932092405255),
    'learning_rate': np.float64(0.2721670232687546),
    'max_depth': 4,
    'n_estimators': 367,
    'subsample': np.float64(0.853239225342979)
}
model = XGBClassifier(**best_params)

with warnings.catch_warnings():
    warnings.simplefilter("ignore")
    results =clf.evaluation_pipeline_save_misclassified(model)
```
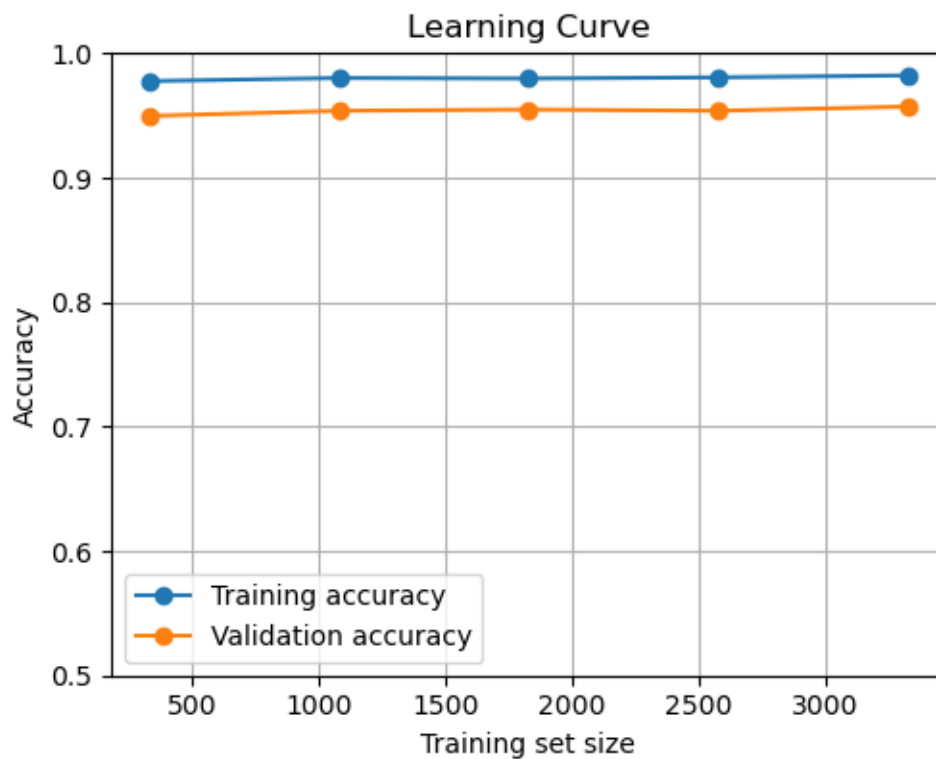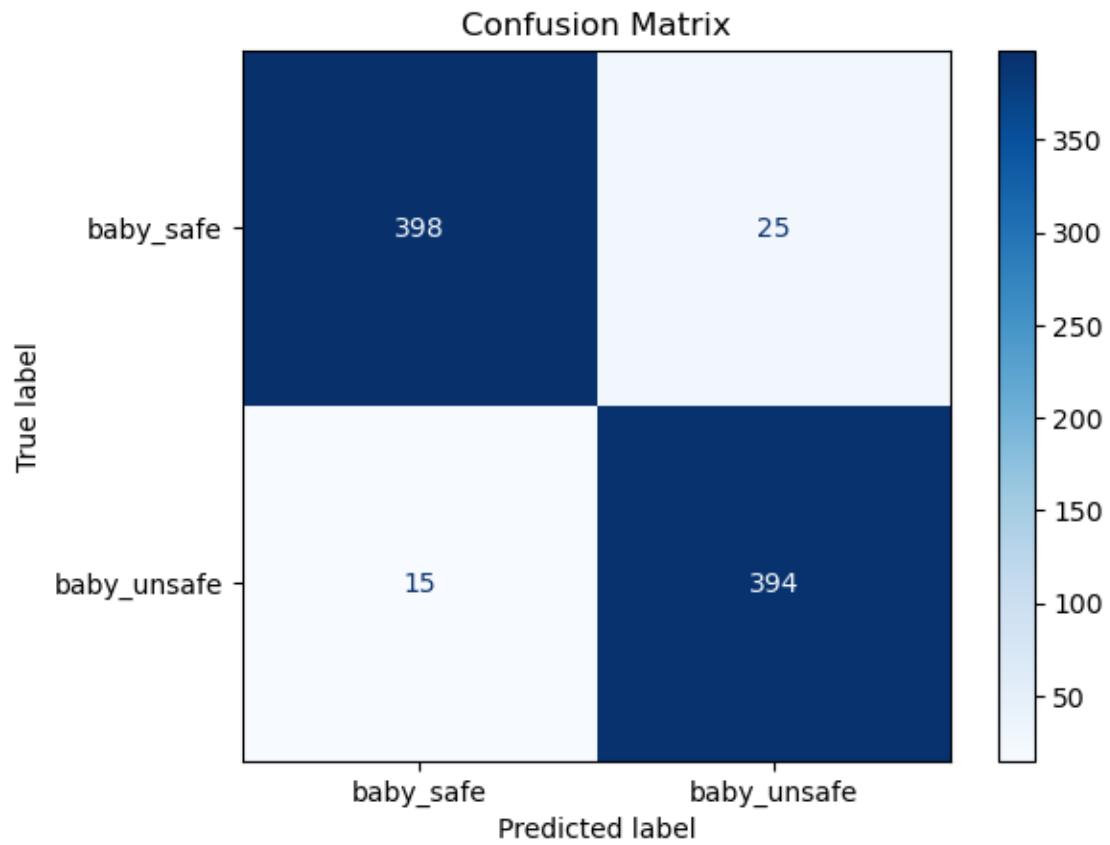
```
--------------------------------------------------------------------------------
----------
-------------------------------------FIRST
ANALYSIS------------------------------------
```

Ordered feature importances

Learning Curve

```
Dataset labels:-------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report---------------------------------------------
              precision    recall  f1-score   support

    baby_safe      0.96      0.94      0.95       423
  baby_unsafe      0.94      0.96      0.95       409

     accuracy                          0.95       832
    macro avg      0.95      0.95      0.95       832
 weighted avg      0.95      0.95      0.95       832

Confusion matrix-----------------------------------
```
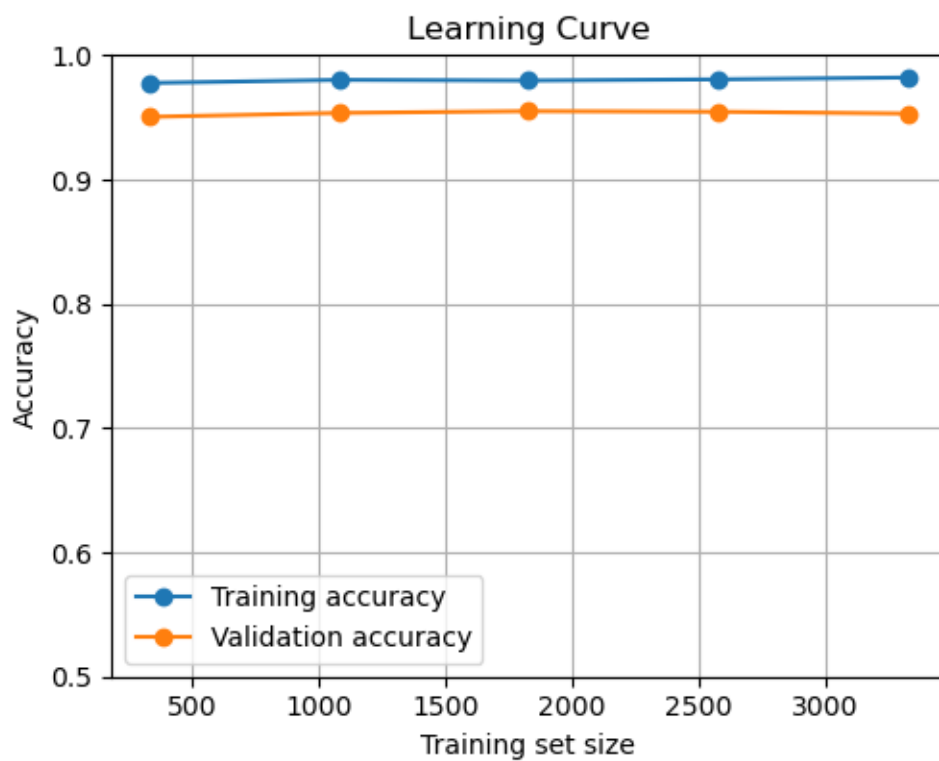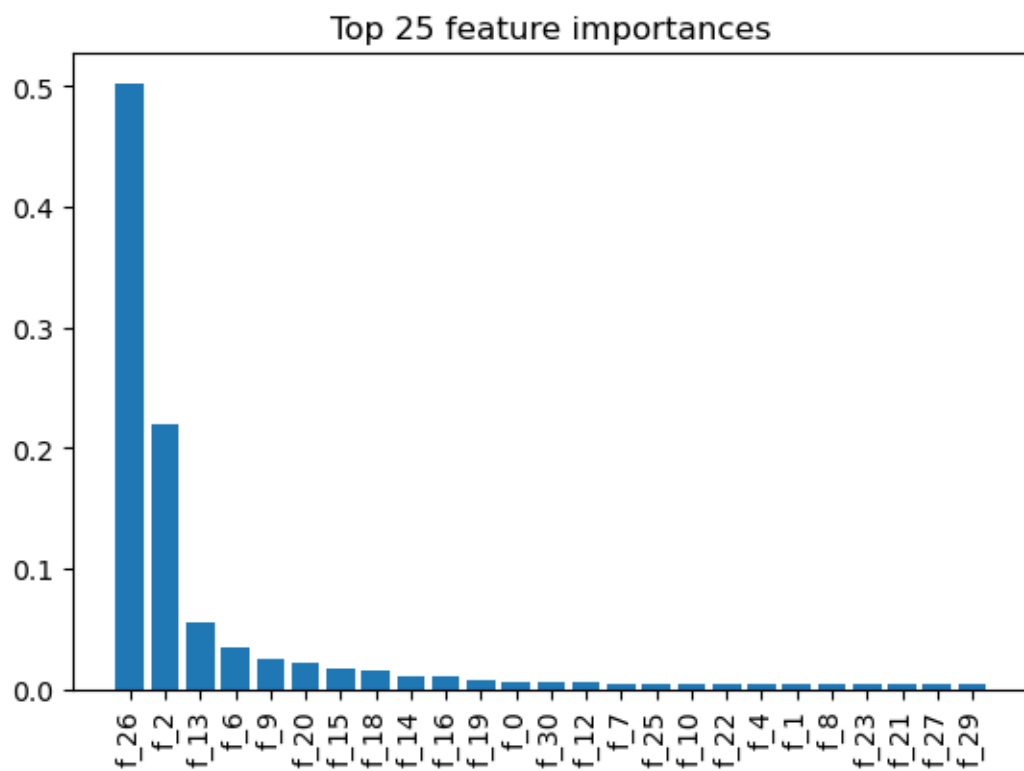
Confusion Matrix

--------------------------------TOP 25 FEATURES
ANALYSIS--------------------------------

Top 25 feature importances

Learning Curve

```
Dataset labels:--------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report-----------------------------------------------
              precision    recall  f1-score   support

   baby_safe       0.98      0.94      0.96       423
 baby_unsafe       0.94      0.98      0.96       409

    accuracy                           0.96       832
   macro avg       0.96      0.96      0.96       832
weighted avg       0.96      0.96      0.96       832

Confusion matrix-------------------------------------
```
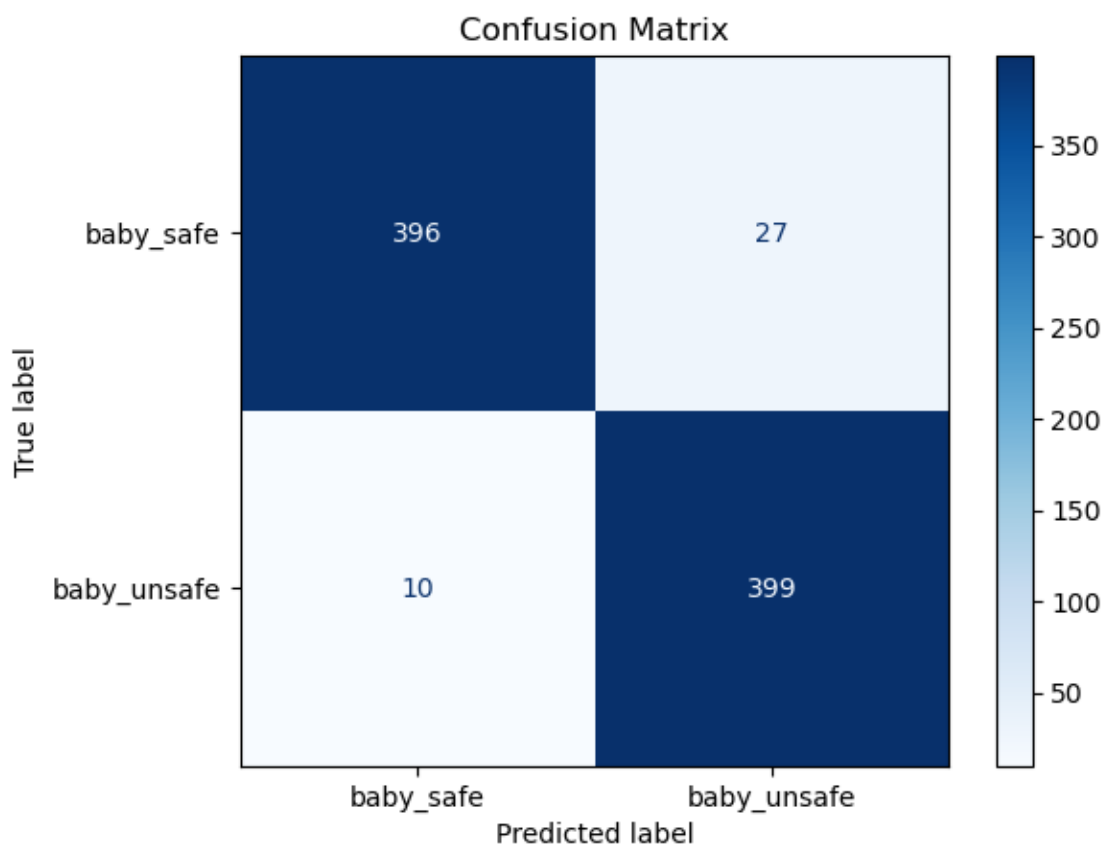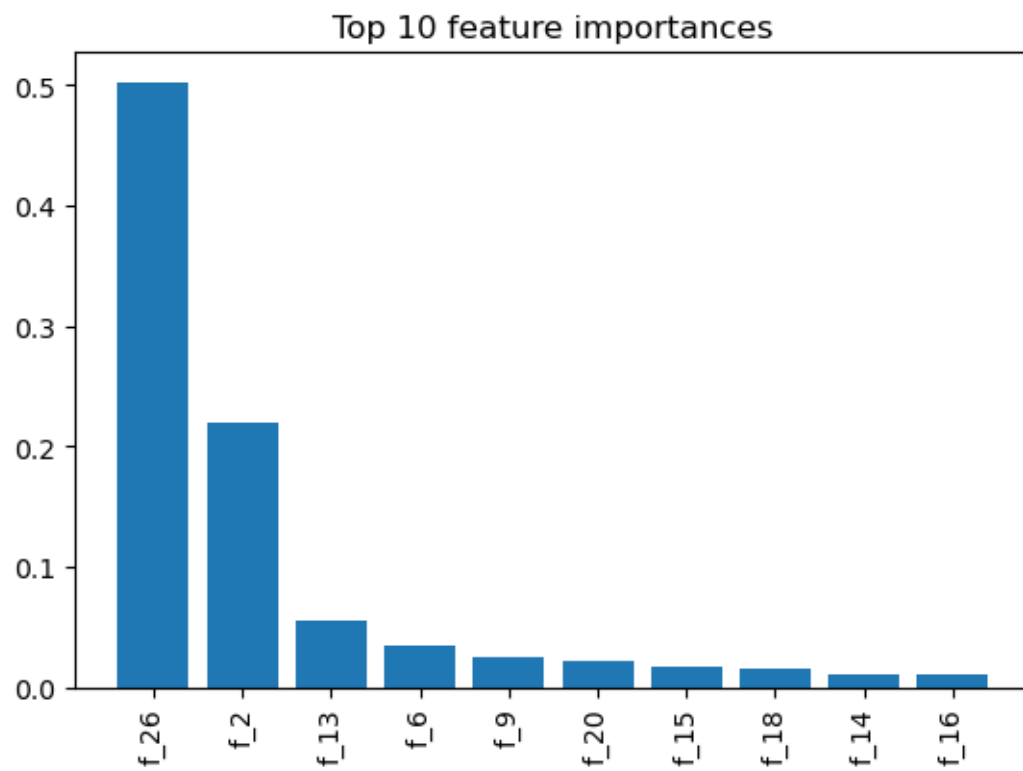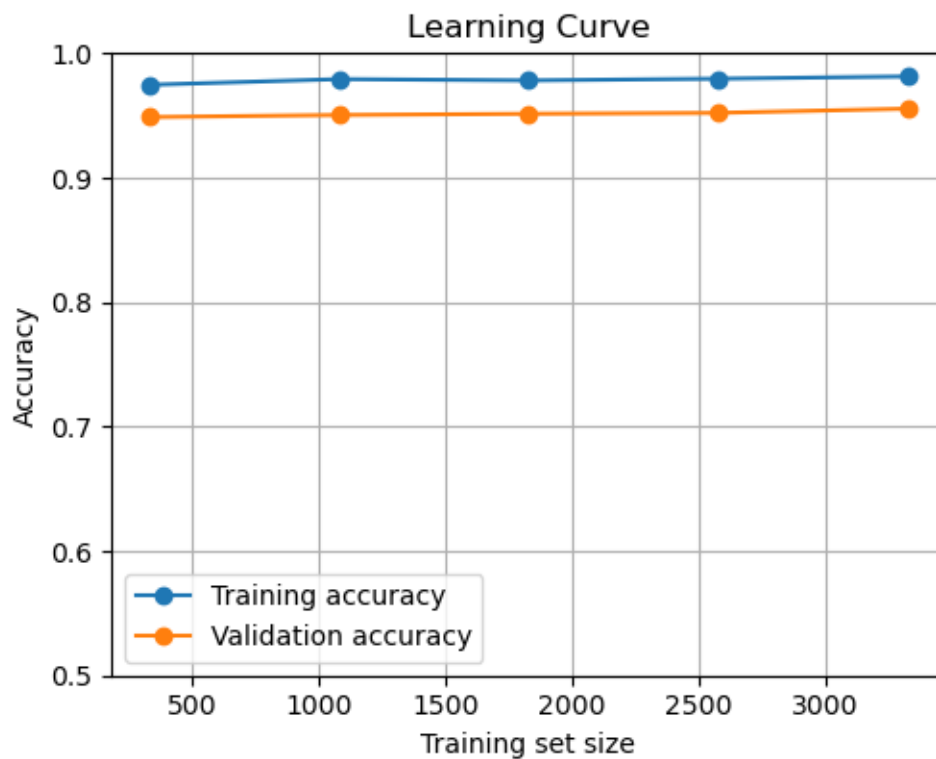
Confusion Matrix

	baby_safe	baby_unsafe
baby_safe	396	27
baby_unsafe	10	399

True label / Predicted label

```
-----------------------------------------------------------------------------
----------
```

--------------------------------TOP 10 FEATURES
ANALYSIS---------------------------------

Top 10 feature importances

Learning Curve

```
Dataset labels:-------------------------------------
{'baby_safe': 0, 'baby_unsafe': 1}

Report----------------------------------------------
              precision    recall  f1-score   support

   baby_safe       0.96      0.95      0.95       423
 baby_unsafe       0.94      0.96      0.95       409

    accuracy                           0.95       832
   macro avg       0.95      0.95      0.95       832
weighted avg       0.95      0.95      0.95       832

Confusion matrix------------------------------------
```
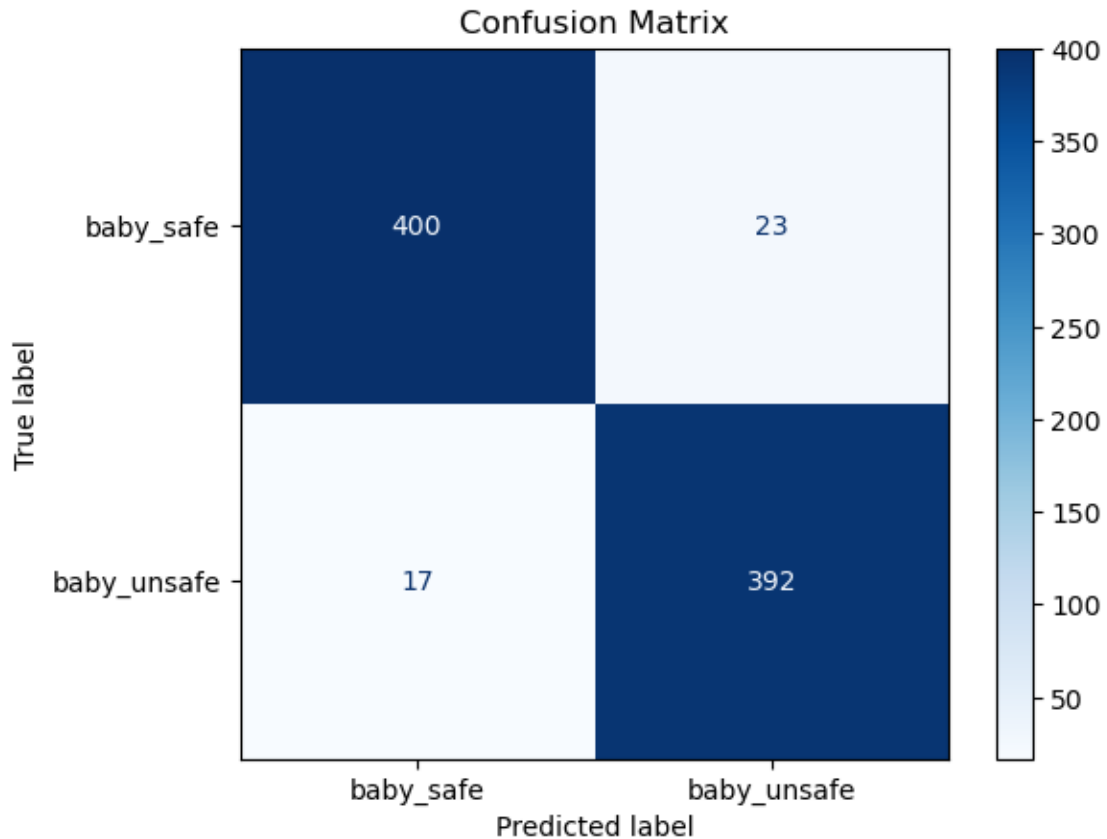
Confusion Matrix


```
[17]: prediction = results["all_features"]["y_predicted"]
      true_y= clf.y_test
      misclassified = np.where(true_y != prediction)[0]

      misclassified_images = [clf.images_paths_test[i] for i in misclassified]
      image_dataset_path=emb_builder.dataset

      folder_path = f"{project_dir}image_prediction/
       ↪approach_supervised_learning_metric_optimized/"
      if not os.path.exists(folder_path):
          os.makedirs(folder_path)

      from PIL import Image, ImageDraw, ImageFont

      for img_path, prediction in zip(misclassified_images, prediction):
              img = Image.open(f"{image_dataset_path}/{img_path}")
              draw = ImageDraw.Draw(img)

              try:
```

```python
            font = ImageFont.truetype("DejaVuSans-Bold.ttf", size=34)  #␣
↪Imposta la dimensione del font
        except IOError:
            font = ImageFont.load_default()  # Usa il font di default se il␣
↪file ttf non è trovato

        predicted_class = [key for key, value in emb_builder.classes_bs.items()␣
↪if value == prediction][0]
        text = f"{predicted_class}"
        text_position = (50, 50)
        text_color = (255, 0, 0)
        draw.text(text_position, text, fill=text_color, font=font)

        new_image_path = f"{folder_path}{img_path}"
        img.save(new_image_path)

print(f"Misclassified images successfully saved in {folder_path}")
```

Misclassified images successfully saved in /home/terra/Desktop/unimore/AI_engine
ering/SIDS_revelation_project/image_prediction/approach_supervised_learning_metr
ic_optimized/

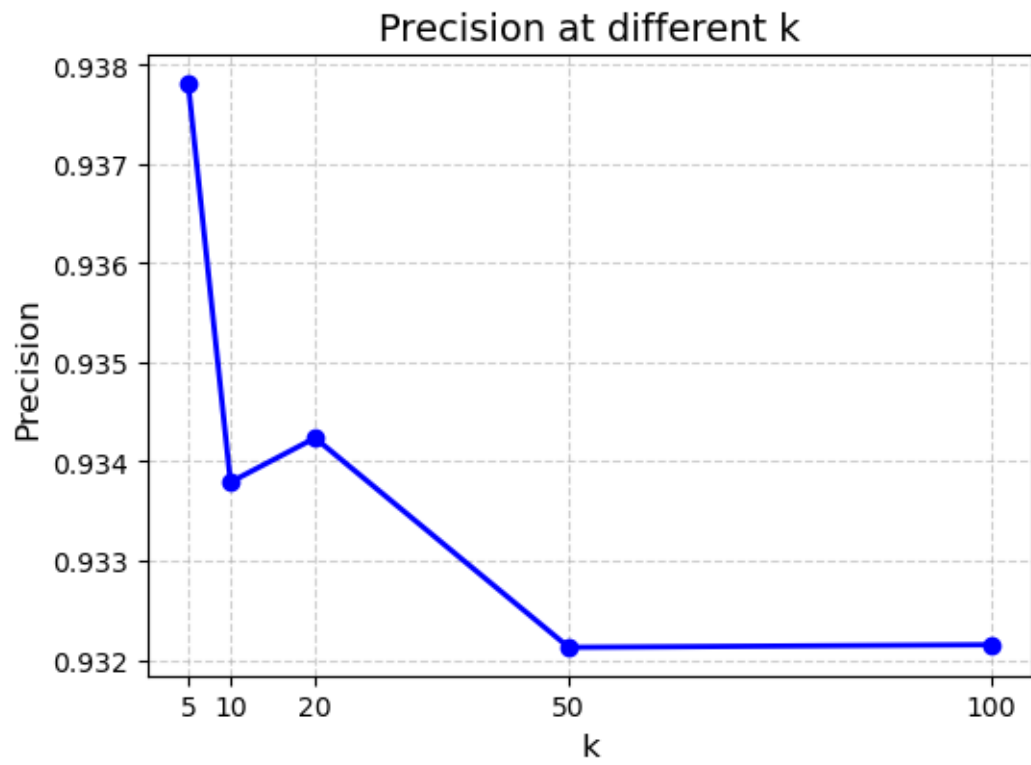Retrieval is invariant, same embeddings as previous point

```python
[18]: ret = ImageRetrieval(embeddings_new, emb_builder.y, emb_builder.image_paths,␣
  ↪emb_builder.dataset, emb_builder.classes_bs)
ret.report('euclidean')
```
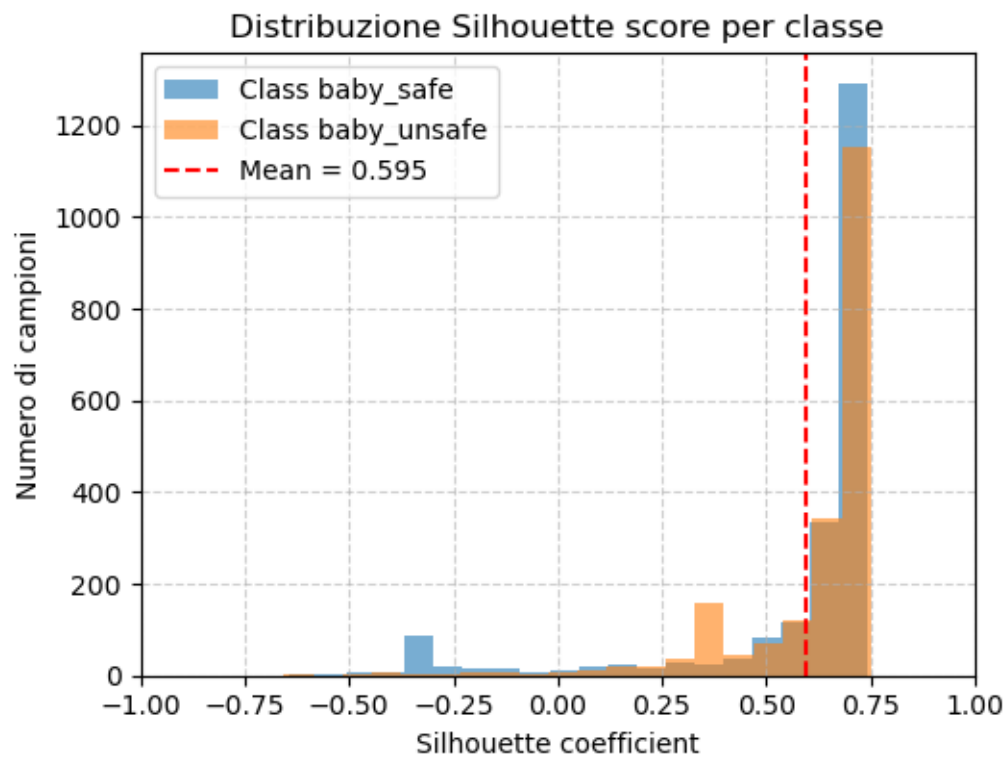
Precision at different
k:--

Precision at different k

Recall at
R--
0.8902544401436857

Silhouette
score--

Distribuzione Silhouette score per classe

Silhouette score (euclidean): 0.595

Embeddings
distributions--

```
[19]:  image_paths = emb_builder.image_paths
       idx_query = 99
       image_to_retrieve = f"{emb_builder.dataset}/{image_paths[idx_query]}"

       print("Image to retrieve")
       img = mpimg.imread(image_to_retrieve)
       plt.figure(figsize=(3, 3))
       plt.imshow(img)
       plt.axis('off')
       plt.show()

       distances_all, image_paths_similar_all = ret.
        ↪retrieve_similar(idx_query=idx_query,k=5,verbose=False)
       ret.show_images(image_paths_similar_all)
```

Image to retrieve

Image 1	Image 2	Image 3	Image 4	Image 5

[ ]: