Prognosis prediction based on estrogen receptor signaling activity from H&E staining by deep learning

In [14]:
```
import sys
print("Python version:", sys.version)
!pip list | grep 'slideio\|pandas\|numpy\|matplotlib\|rpy2\|torch\|torchv
import slideio
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image
import os
import math
import rpy2
import json
import torch
torch.manual_seed(0)
from torch.utils.data import DataLoader, Dataset
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import torch.backends.cudnn as cudnn
import torchvision
from torchvision.models import ResNet50_Weights
from torchvision import datasets, models, transforms
import torchvision.transforms.functional as F
from PIL import Image
import matplotlib.pyplot as plt
import time
import os
import copy
from sklearn.model_selection import train_test_split
import scipy
from sklearn.model_selection import KFold
from sklearn.metrics import roc_auc_score, roc_curve, RocCurveDisplay
from scipy.stats import spearmanr, pearsonr
import kaplanmeier as km
import rpy2.robjects as ro
from rpy2.robjects.packages import importr
from rpy2.robjects import pandas2ri
import rpy2.robjects as robjects
from rpy2.robjects.conversion import localconverter
rinstalled = robjects.globalenv.find("installed.packages")
rversion = robjects.globalenv.find("R.Version")
rpkgs = rinstalled()
rvers = rpkgs.rx(robjects.StrVector(["GSVA"]), robjects.StrVector(["Versi
print(rversion().rx2("version.string"))
print("GSVA Version:", rvers)

def gsva(geneExpressionProfile, gene_sets):
    rbase = importr('base')
```

```
    print("Converting df")
    with localconverter(ro.default_converter + pandas2ri.converter):
        geneExpressionProfile_r = ro.conversion.py2rpy(geneExpressionProf
    gene_sets_r = ro.ListVector(gene_sets)
    gsvar = importr("GSVA")
    es = gsvar.gsva(rbase.as_matrix(geneExpressionProfile_r), gene_sets_r
    es_df = pd.DataFrame(np.array(es.transpose()), index=es.colnames, col
    return es_df

def finding_best_threshold_with_FS_SS(fs_eeres_df, ss_eeres_df, survival_
    bestp = 1
    bestq = 0
    for q in np.arange(q01, q09, 0.01):
        result1 = km.fit(fs_eeres_df[f'{survival_type[0]}_MONTHS'], fs_ee
        result2 = km.fit(ss_eeres_df[f'{survival_type[1]}_MONTHS'], ss_ee
        average_p = (result1["logrank_P"]+result2["logrank_P"])/2
        if (result1["logrank_P"]<0.05 or result2["logrank_P"]<0.05) and a
            bestp=average_p
            bestq=q
    if bestp==1:
        print('not significant')
    else:
        print(f"Best {value_name} threshold:", bestq)
        result = km.fit(fs_eeres_df[f'{survival_type[0]}_MONTHS'], fs_eer
        km.plot(result, title=f"{survival_type[0]} of ER+/HER2- stratifie
        plt.show()
        result = km.fit(ss_eeres_df[f'{survival_type[1]}_MONTHS'], ss_eer
        km.plot(result, title=f"{survival_type[1]} of ER+/HER2- stratifie
        plt.show()
```

```
Python version: 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0]
kaplanmeier               0.1.8
matplotlib                3.5.1
matplotlib-inline         0.1.6
matplotlib-venn           0.11.9
numpy                     1.24.1
pandas                    1.5.2
rpy2                      3.5.7
scipy                     1.8.0
sklearn                   0.0.post1
slideio                   2.0.0
torch                     1.13.1
torchaudio                0.13.1
torchvision               0.14.1
[1] "R version 4.2.3 (2023-03-15)"

GSVA Version: [1] "1.46.0"
```

Image resizing, annotating and cropping

In [ ]:
```
meta = pd.read_table('File_metadata.txt', encoding='utf-8')
files = ("Images/" + meta_nondup["file_id"] + '/' + meta_nondup["file_nam
for file in files:
    print(file)
    slide = slideio.open_slide(file,'SVS')
    print("slide")
    num_scenes = slide.num_scenes
    print("num")
    scene = slide.get_scene(0)
    print("scene")
    rect = scene.rect
    print(rect)
    size=1024
    image = scene.read_block(size=(int(size*rect[2]/rect[3]) if rect[2]>r
    image_tr = scipy.ndimage.rotate(image, 180)
    print("image")
    block1 = image[0:size, 0:size, 0:3]
    block2 = image_tr[0:size, 0:size, 0:3]
    matplotlib.image.imsave(file+"-1.jpg", block1)
    matplotlib.image.imsave(file+"-2.jpg", block2)
    matplotlib.image.imsave(file+".jpg", image)
    print("save")
```

Loading TCGA Pan-Cancer BRCA level 3 gene expression data from cBioPortal

In [6]:
```
brca_lv3 = pd.read_csv("brca_tcga_pan_can_atlas_2018/data_mrna_seq_v2_rse
brca_lv3.index = [i[:12] for i in brca_lv3.index]
brca_lv3
```

Out[6]:

Hugo_Symbol	A1BG	A1CF	A2BP1	A2LD1	A2M	A2M-AS1	A2ML1	A4GALT
TCGA-3C-AAAU	197.090	0.0000	0.0000	102.9630	5798.37	32.2187	1.3786	68.2424
TCGA-3C-AALI	237.384	0.0000	0.0000	70.8646	7571.98	29.9782	4.3502	157.6940
TCGA-3C-AALJ	423.237	0.9066	0.0000	161.2600	8840.40	17.2620	0.0000	573.8890
TCGA-3C-AALK	191.018	0.0000	0.0000	62.5072	10960.20	17.8527	1.6549	506.4130
TCGA-4H-AAAK	268.881	0.4255	3.8298	154.3700	9585.44	31.5787	3.4043	342.1280
...	...	...	...	...	...	...	...	...
TCGA-WT-AB44	471.285	0.0000	0.0000	61.7308	5409.31	39.6823	6.5160	356.7500
TCGA-XX-A899	223.220	0.0000	0.3937	131.2280	20348.80	27.2283	0.3937	505.5120
TCGA-XX-A89A	255.135	2.3618	1.4171	79.9291	17094.80	31.7572	55.7393	615.4940
TCGA-Z7-A8R5	439.543	0.0000	0.5973	81.3010	36838.50	84.0964	2.3893	456.3510
TCGA-Z7-A8R6	248.327	0.0000	0.0000	25.1866	7339.17	8.3723	4.1757	768.6820

1082 rows × 20511 columns

Early Estrogen Response Enrichment Scores (EERES) by GSVA

In [3]:

```
h_gs = json.loads(open("h.all.v2022.1.Hs.json", 'r').read())
early_es_gs = {'HALLMARK_ESTROGEN_RESPONSE_EARLY': h_gs['HALLMARK_ESTROGE
# brca_earlyes_es = gsva((brca_lv3+1).applymap(math.log2).transpose(), ea
# brca_earlyes_es.index = [i[:12] for i in brca_earlyes_es.index]
# brca_earlyes_es.to_csv("Table S1.csv")
brca_earlyes_es = pd.read_csv("Table S1.csv", index_col=0)
brca_earlyes_es
```

Out[3]:

	HALLMARK_ESTROGEN_RESPONSE_EARLY
TCGA-3C-AAAU	0.001404
TCGA-3C-AALI	-0.409362
TCGA-3C-AALJ	-0.038938
TCGA-3C-AALK	0.356928
TCGA-4H-AAAK	0.154434
...	...
TCGA-WT-AB44	-0.224513
TCGA-XX-A899	0.161219
TCGA-XX-A89A	-0.122835
TCGA-Z7-A8R5	-0.193181
TCGA-Z7-A8R6	0.209939

1082 rows × 1 columns

Loading TCGA BRCA IHC data from GDC Portal

In [7]:
```
meta_erihc = pd.read_table("nationwidechildrens.org_clinical_patient_brca
meta_erihc

# Joining IHC data with ESR1/EERES data
eeres_ihc_df = brca_earlyes_es.join(meta_erihc, how='inner')
eeres_ihc_df["Subtype"] = None
eeres_ihc_df["Subtype"][(eeres_ihc_df["er_status_by_ihc"]=='Positive') &
eeres_ihc_df["Subtype"][(eeres_ihc_df["er_status_by_ihc"]=='Negative') &
eeres_ihc_df_erposerbb2neg = eeres_ihc_df[eeres_ihc_df["Subtype"]=="ER+/H
eeres_ihc_df_erposerbb2neg
eeres_ihc_df_tnbc = eeres_ihc_df[eeres_ihc_df["Subtype"]=="TNBC"]
eeres_ihc_df["EERES"] = brca_earlyes_es.loc[eeres_ihc_df.index]
eeres_ihc_df["ESR1"] = brca_lv3.loc[eeres_ihc_df.index]["ESR1"]
eeres_ihc_df
```

```
/tmp/ipykernel_93586/3367593681.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  eeres_ihc_df["Subtype"][(eeres_ihc_df["er_status_by_ihc"]=='Positive')
& (eeres_ihc_df["her2_status_by_ihc"]=='Negative')] = "ER+/HER2-"
/tmp/ipykernel_93586/3367593681.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  eeres_ihc_df["Subtype"][(eeres_ihc_df["er_status_by_ihc"]=='Negative')
& (eeres_ihc_df["her2_status_by_ihc"]=='Negative') & (eeres_ihc_df["pr_s
tatus_by_ihc"]=='Negative')] = "TNBC"
```

Out[7]:

	HALLMARK_ESTROGEN_RESPONSE_EARLY	her2_status_by_ihc	er_status_by_ihc	pr
TCGA-3C-AAAU	0.001404	Negative	Positive	
TCGA-3C-AALI	-0.409362	Positive	Positive	
TCGA-3C-AALJ	-0.038938	Indeterminate	Positive	
TCGA-3C-AALK	0.356928	Positive	Positive	
TCGA-4H-AAAK	0.154434	Equivocal	Positive	
...	...	...	...	...
TCGA-WT-AB44	-0.224513	Negative	Positive	
TCGA-XX-A899	0.161219	Negative	Positive	
TCGA-XX-A89A	-0.122835	Negative	Positive	
TCGA-Z7-A8R5	-0.193181	Negative	Positive	
TCGA-Z7-A8R6	0.209939	Negative	Positive	

1081 rows × 7 columns

Loading file metadata and file locations

In [8]:
```
meta = pd.read_table('File_metadata.txt', encoding='utf-8')
meta_nondup = meta[~meta['cases.0.submitter_id'].duplicated(keep='first')
meta_nondup = meta_nondup.set_index('cases.0.submitter_id') # Get images
files = ("Images/" + meta_nondup["file_id"] + '/' + meta_nondup["file_nam
meta_nondup.to_csv("Table S2.csv")
meta_nondup
```

Out[8]:

	cases.0.case_id	cases.0.samples.0.sample_id	cases.0.samples.0.sam
cases.0.submitter_id			
TCGA-AO-A1KQ	d2219a3f-4c55-4dee-9b08-	38c306c0-bd3f-41b6-bf2e-	Prima

	c6f330dd1acf	c27f6df81567	
TCGA-D8-A1XU	332148f5-f070-4c20-8eb1-4d8c0673aa52	ba37ce51-fb89-4e7a-964c-e69538d69e66	Prima
TCGA-B6-A0IK	0491063c-14a7-4104-8002-1459126332f3	e84a2787-e329-4db4-b636-2b94ffc0289d	Prima
TCGA-BH-A18L	9ddc3e7b-8b54-4a83-8335-8053940f56c1	d5dc0fec-ff5e-4dea-82f6-a25f52342f94	Prima
TCGA-E9-A247	636b0a90-c8c2-4ad6-a77d-6cd428a826d1	d4054576-335d-4e27-b3a0-d041f07cd393	Prima
...	...	...	
TCGA-A2-A0T7	43d5632e-7272-4aad-9496-7f5a99671223	a79f80f7-5704-4eae-856b-9b3fd0ba26f1	Prima
TCGA-E9-A1RC	5cdae21d-eee5-478f-932a-0f51fcf5f031	d3393b43-a688-4de3-9d1f-11b76f69e32a	Prima
TCGA-A2-A4RY	5a53ec47-e63f-44a2-8078-87659544fb04	e742341b-0392-4f84-89c6-97264a235b2b	Prima
TCGA-AC-A3QP	6f4c8d30-47fb-47df-9eb7-4e5881e3711e	bbcb581f-1f71-4335-a552-e0eb49c8246a	Prima
TCGA-EW-A1IW	5e2cbc56-97f9-4b94-ade2-a3f739bcc7be	665bdee5-6983-45c8-b09f-e1898d65a380	Prima

1062 rows × 11 columns

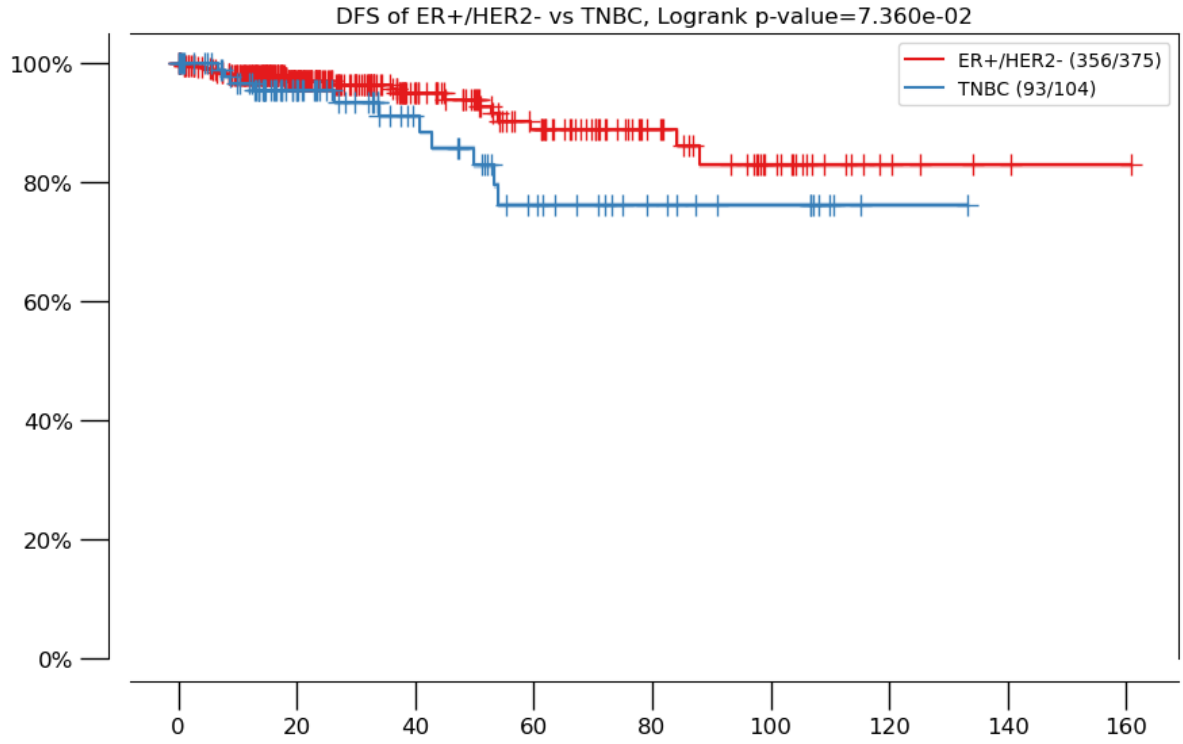
Loading survival data from cBioPortal

In [9]:
```
brca_clinical = pd.read_table("brca_tcga_pan_can_atlas_2018/data_clinical
brca_clinical = brca_clinical[["OS_STATUS", "OS_MONTHS","PFS_STATUS", "PF
brca_clinical["OS_STATUS"] = [s[0] for s in brca_clinical["OS_STATUS"]]
brca_clinical["PFS_STATUS"] = [str(s)[0] for s in brca_clinical["PFS_STAT
brca_clinical["DFS_STATUS"] = [str(s)[0] for s in brca_clinical["DFS_STAT
brca_clinical["DSS_STATUS"] = [s[0] if str(s) != 'nan' else np.nan for s
```

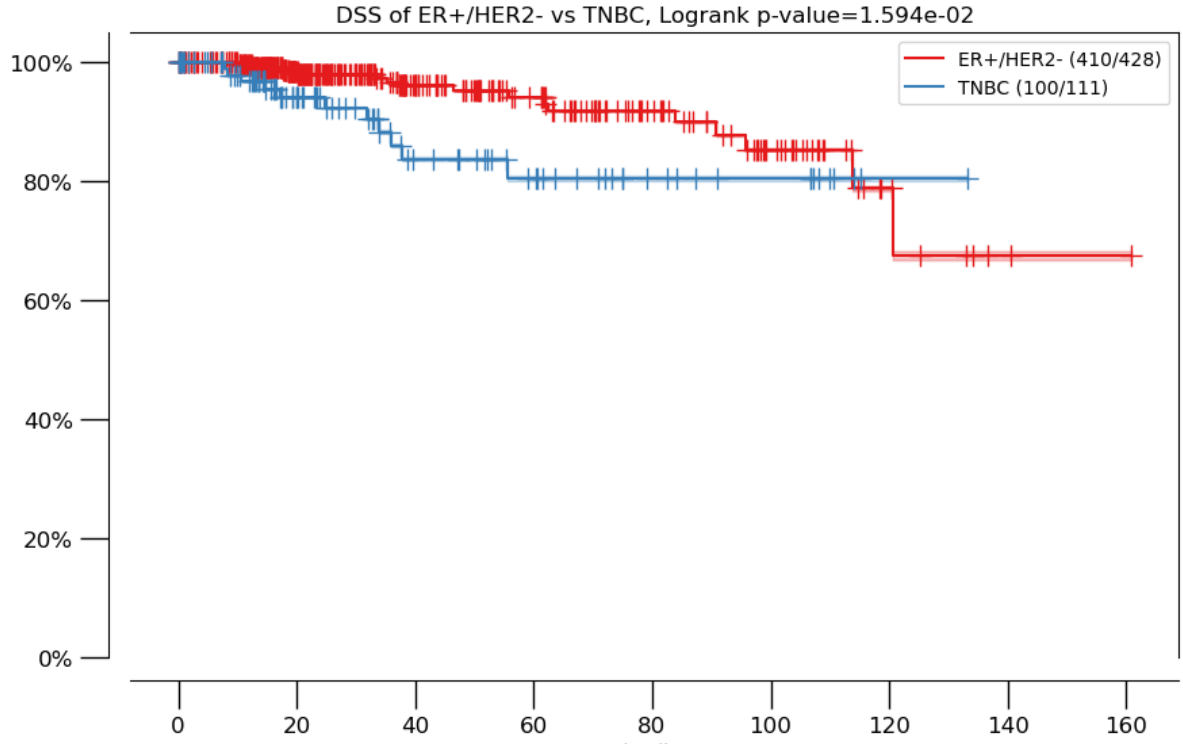
Joining survival data with EERES_IHC data

Show the survival of ER+/HER2- and TNBC of all data

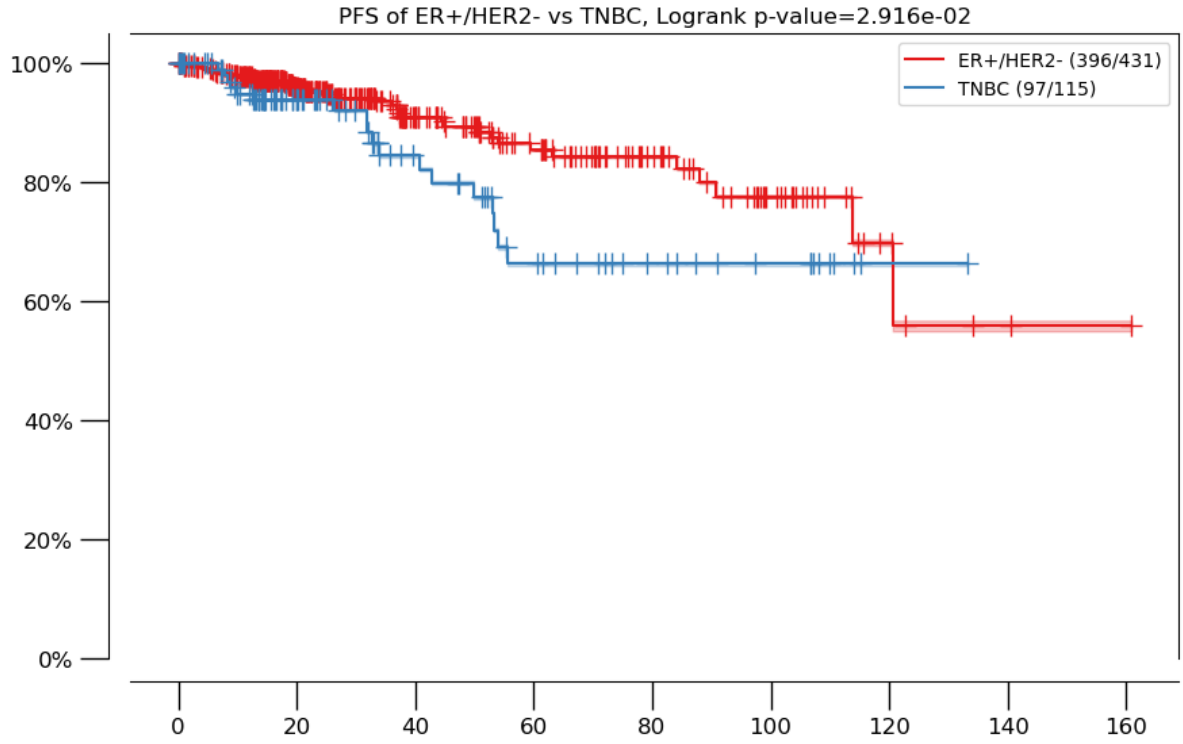
In [10]:
```
brca_clinical_eeres_ihc_df = brca_clinical.join(eeres_ihc_df, how='inner'

brca_clinical_dfs = brca_clinical_eeres_ihc_df[["DFS_MONTHS", "DFS_STATUS
brca_clinical_dss = brca_clinical_eeres_ihc_df[["DSS_MONTHS", "DSS_STATUS
brca_clinical_pfs = brca_clinical_eeres_ihc_df[["PFS_MONTHS", "PFS_STATUS
brca_clinical_os = brca_clinical_eeres_ihc_df[["OS_MONTHS", "OS_STATUS",


result = km.fit(brca_clinical_dfs['DFS_MONTHS'], brca_clinical_dfs['DFS_S
km.plot(result, title=f"DFS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_dss['DSS_MONTHS'], brca_clinical_dss['DSS_S
km.plot(result, title=f"DSS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_pfs['PFS_MONTHS'], brca_clinical_pfs['PFS_S
km.plot(result, title=f"PFS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_os['OS_MONTHS'], brca_clinical_os['OS_STATU
km.plot(result, title=f"OS of ER+/HER2- vs TNBC, Logrank p-value={result[
plt.show()
plt.scatter(eeres_ihc_df[eeres_ihc_df["Subtype"]=="ER+/HER2-"]["ESR1"].ap
# plt.scatter(eeres_ihc_df[eeres_ihc_df["Subtype"]=="ERBB2+"]["esr1"].app
plt.scatter(eeres_ihc_df[eeres_ihc_df["Subtype"]=="TNBC"]["ESR1"].apply(m


plt.legend(["ER+/HER2-", "TNBC"])
plt.xlabel("ESR1")
plt.ylabel("EERES")
plt.show()
```



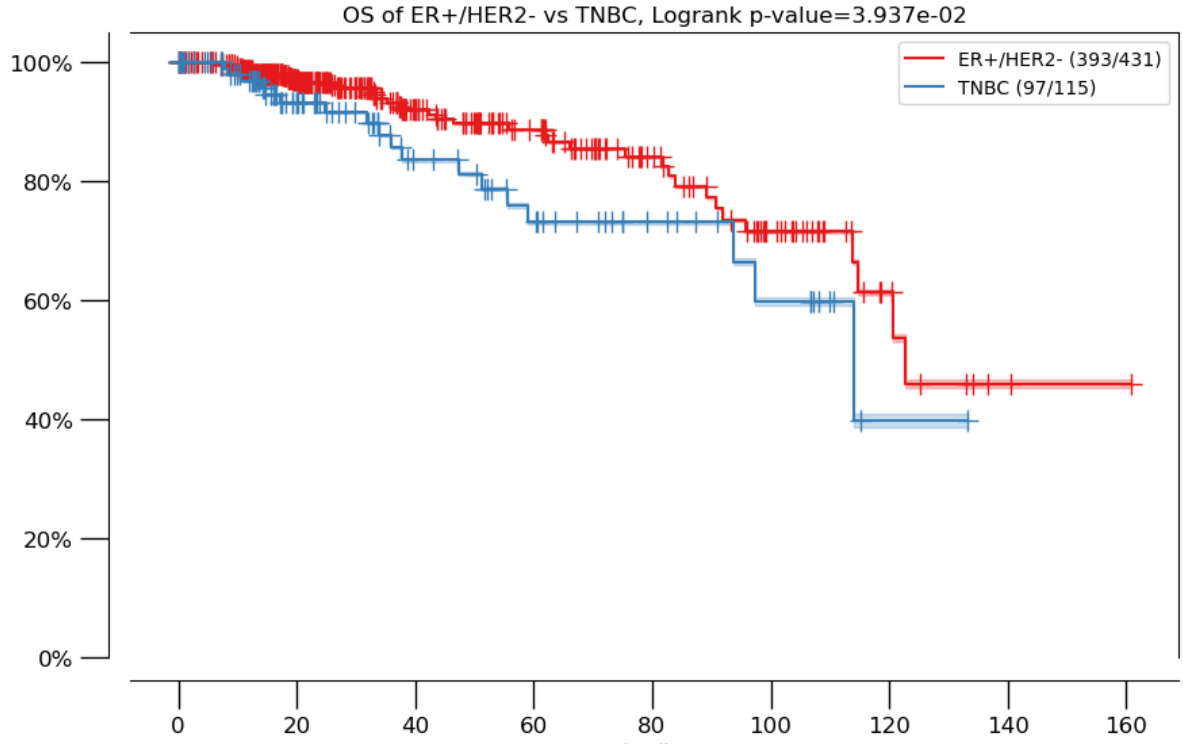
ER+/HER2- (356/375)									
At risk	370	210	105	66	35	18	5	2	1
Censored	5	156	258	292	323	338	351	354	355
Events	0	9	12	17	17	19	19	19	19
TNBC (93/104)									
At risk	104	58	34	21	12	8	1	0	0
Censored	0	42	64	72	81	85	92	93	93
Events	0	4	6	11	11	11	11	11	11



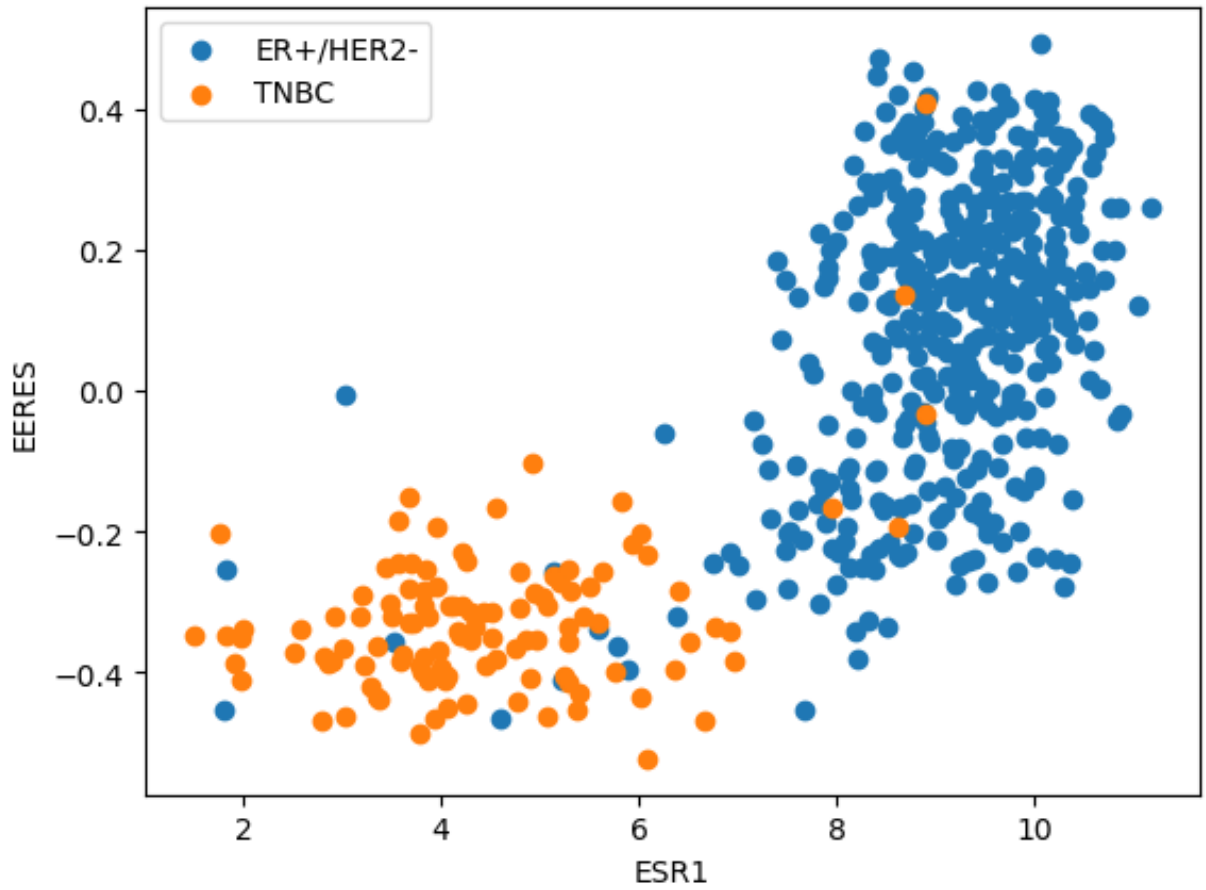
ER+/HER2- (410/428)									
At risk	422	247	130	88	53	28	8	2	1
Censored	6	176	289	329	362	384	403	408	409
Events	0	5	9	11	13	16	17	18	18
TNBC (100/111)									
At risk	111	61	34	24	13	9	1	0	0
Censored	0	45	67	76	87	91	99	100	100
Events	0	5	10	11	11	11	11	11	11



ER+/HER2- (396/431)									
At risk	425	243	125	80	47	23	6	2	1
Censored	6	174	283	322	354	375	391	394	395
Events	0	14	23	29	30	33	34	35	35
TNBC (97/115)									
At risk	115	63	37	23	14	9	1	0	0
Censored	0	46	67	74	83	88	96	97	97
Events	0	6	11	18	18	18	18	18	18



ER+/HER2- (393/431)									
At risk	425	250	131	89	54	29	9	2	1
Censored	6	171	281	319	350	368	386	391	392
Events	0	10	19	23	27	34	36	38	38
TNBC (97/115)									
At risk	115	64	37	26	15	9	1	0	0
Censored	0	45	67	74	85	89	96	97	97
Events	0	6	11	15	15	17	18	18	18



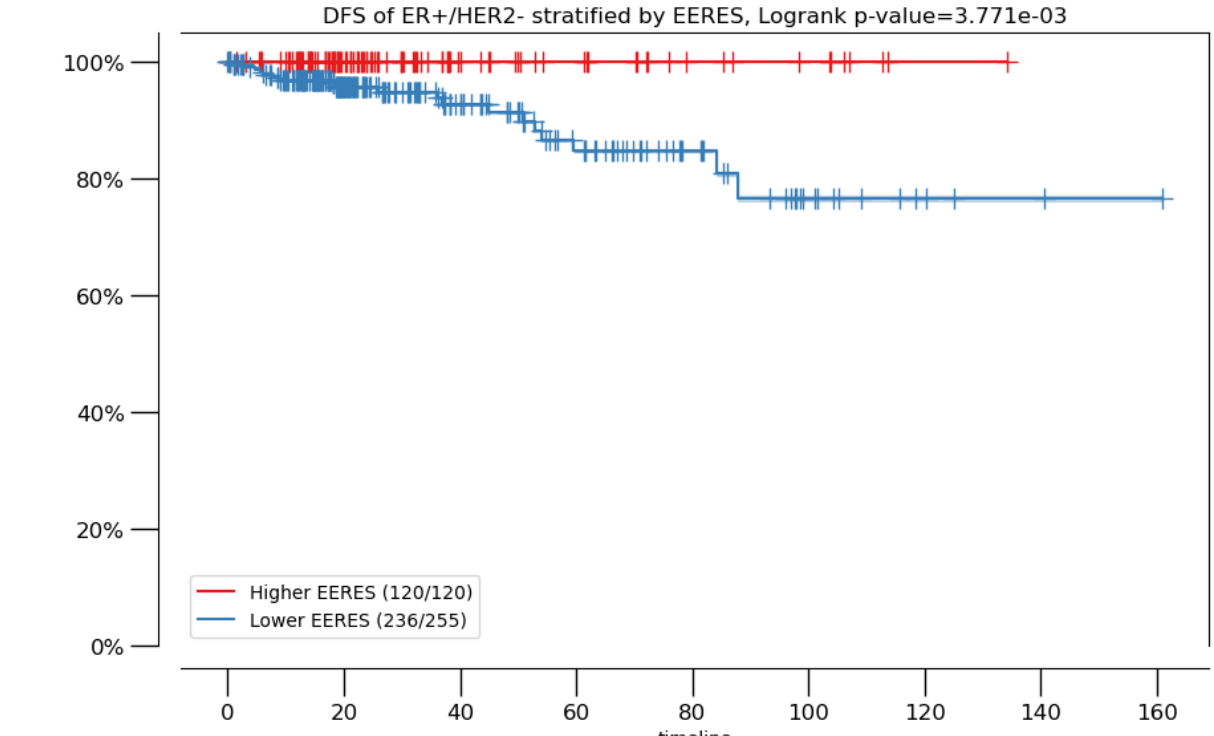
Getting the best thershold of EERES for ER+/HER2- and TNBC patients

In [11]:
```
brca_clinical_erposerbb2neg = brca_clinical_eeres_ihc_df[brca_clinical_ee

brca_clinical_erposerbb2neg_dfs = brca_clinical_erposerbb2neg[["DFS_MONTH
brca_clinical_erposerbb2neg_dss = brca_clinical_erposerbb2neg[["DSS_MONTH
brca_clinical_erposerbb2neg_pfs = brca_clinical_erposerbb2neg[["PFS_MONTH
brca_clinical_erposerbb2neg_os = brca_clinical_erposerbb2neg[["OS_MONTHS"
```

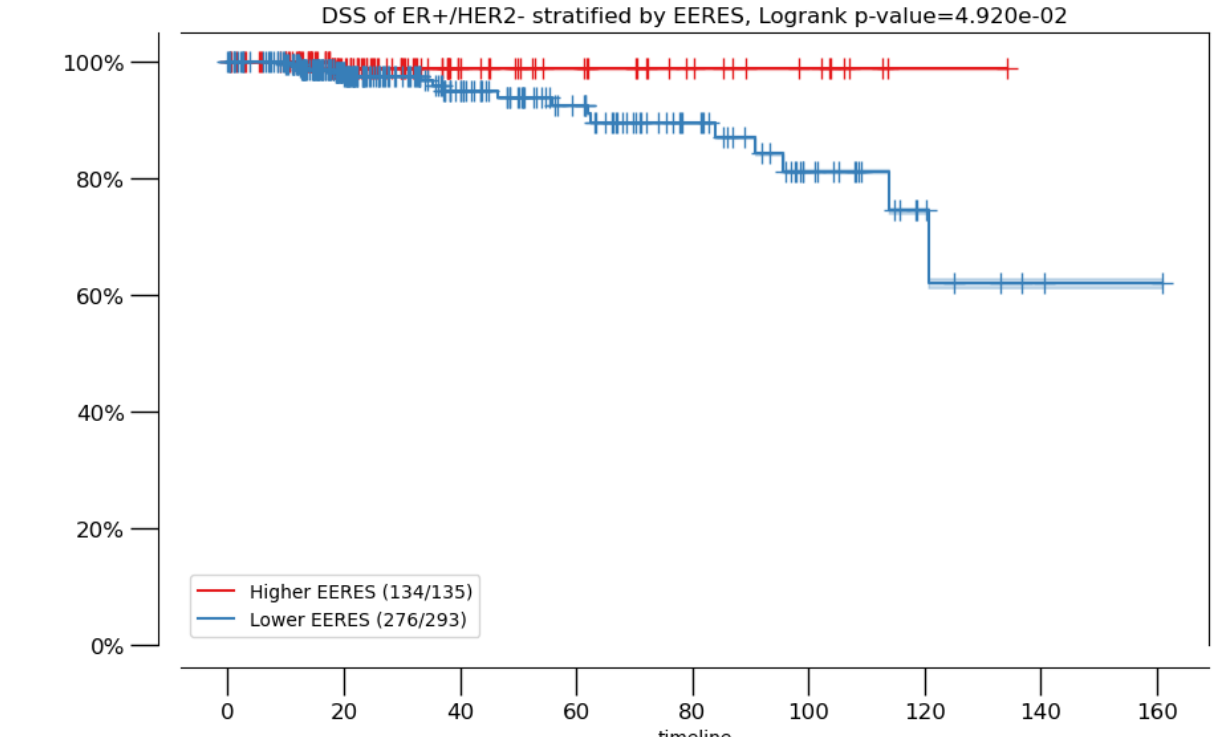
ER+/HER2-

In [15]:
```
print("DFS/DSS")
finding_best_threshold_with_FS_SS(brca_clinical_erposerbb2neg_dfs, brca_c
print("PFS/OS")
finding_best_threshold_with_FS_SS(brca_clinical_erposerbb2neg_pfs, brca_c
```

```
DFS/DSS
Best HALLMARK_ESTROGEN_RESPONSE_EARLY threshold: 0.2020931077758909
```



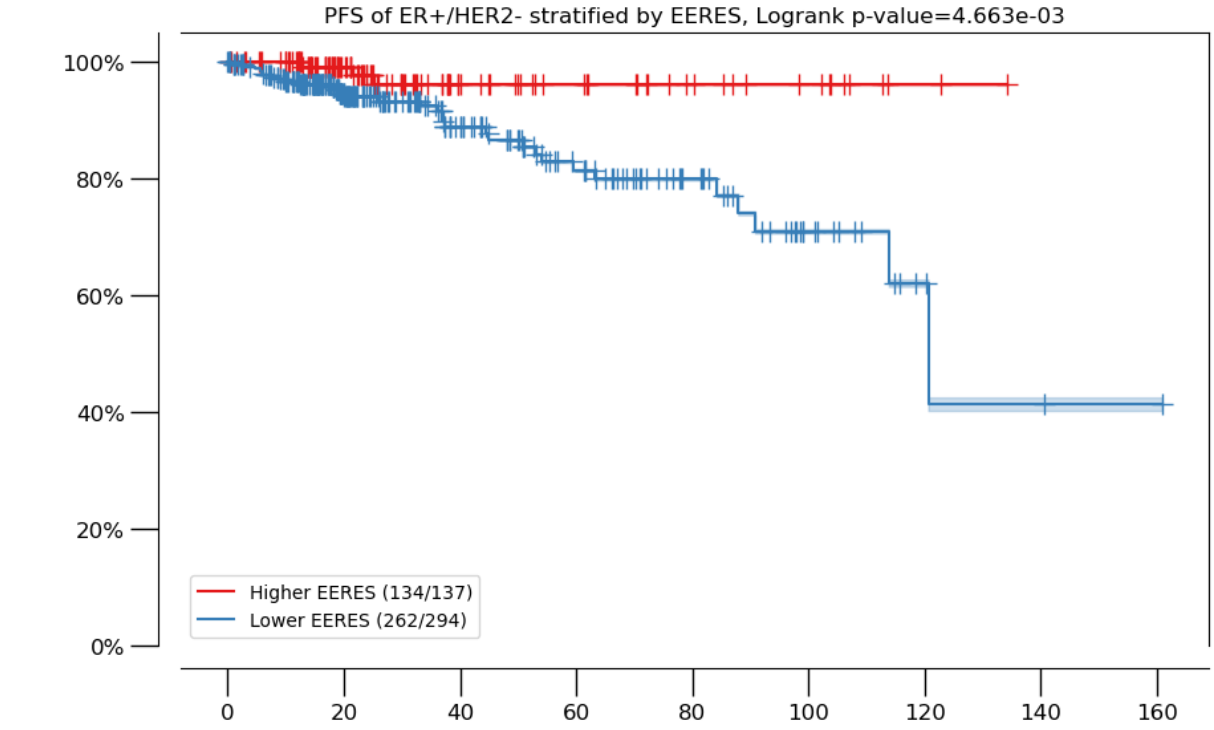
Higher EERES (120/120)									
At risk	120	69	28	19	10	7	1	0	0
Censored	0	51	92	101	110	113	119	120	120
Events	0	0	0	0	0	0	0	0	0
Lower EERES (236/255)									
At risk	250	141	77	47	25	11	4	2	1
Censored	5	105	166	191	213	225	232	234	235
Events	0	9	12	17	17	19	19	19	19



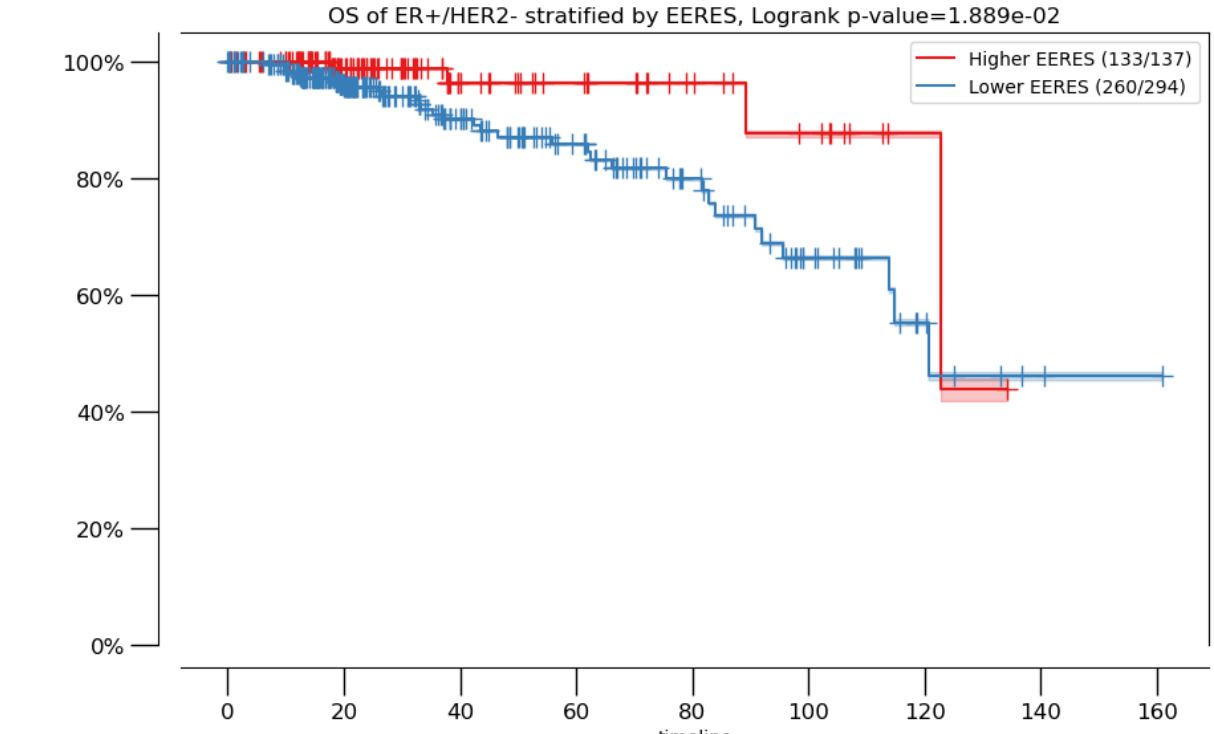
Higher EERES (134/135)									
At risk	135	77	32	22	13	8	1	0	0
Censored	0	57	102	112	121	126	133	134	134
Events	0	1	1	1	1	1	1	1	1
Lower EERES (276/293)									
At risk	287	170	98	66	40	20	7	2	1
Censored	6	119	187	217	241	258	270	274	275
Events	0	4	8	10	12	15	16	17	17

PFS/OS

Best HALLMARK_ESTROGEN_RESPONSE_EARLY threshold: 0.2020931077758909



Higher EERES (134/137)									
At risk	137	79	33	23	14	9	2	0	0
Censored	0	57	101	111	120	125	132	134	134
Events	0	1	3	3	3	3	3	3	3
Lower EERES (262/294)									
At risk	288	164	92	57	33	14	4	2	1
Censored	6	117	182	211	234	250	259	260	261
Events	0	13	20	26	27	30	31	32	32



Higher EERES (133/137)									
At risk	137	79	33	23	14	9	2	0	0
Censored	0	57	102	112	121	125	132	133	133
Events	0	1	2	2	2	3	3	4	4
Lower EERES (260/294)									
At risk	288	171	98	66	40	20	7	2	1
Censored	6	114	179	207	229	243	254	258	259
Events	0	9	17	21	25	31	33	34	34

TNBC

In [16]:
```
brca_clinical_tnbc = brca_clinical_eeres_ihc_df[brca_clinical_eeres_ihc_d

brca_clinical_tnbc_dfs = brca_clinical_tnbc[["DFS_MONTHS", "DFS_STATUS",
brca_clinical_tnbc_dss = brca_clinical_tnbc[["DSS_MONTHS", "DSS_STATUS",
brca_clinical_tnbc_pfs = brca_clinical_tnbc[["PFS_MONTHS", "PFS_STATUS",
brca_clinical_tnbc_os = brca_clinical_tnbc[["OS_MONTHS", "OS_STATUS", 'HA

print("DFS/DSS")
finding_best_threshold_with_FS_SS(brca_clinical_tnbc_dfs, brca_clinical_t
print("PFS/OS")
finding_best_threshold_with_FS_SS(brca_clinical_tnbc_pfs, brca_clinical_t
```

```
DFS/DSS
not significant
PFS/OS
not significant
```

Preparing trainig and testing data

In [17]:

```
# Data augmentation and normalization for training

class MyData_train(Dataset):
    def __init__(self, X, y):
        self.X = X
        self.classes = np.unique(y)
        self.y = np.array(y)
        self.tumorid = y.index

    def __len__(self):
        return len(self.X)

    def __getitem__(self, index):
        transform = transforms.Compose([
                                        transforms.RandomRotation(360),
                                        transforms.RandomHorizontalFlip()
                                        transforms.RandomVerticalFlip(),
                                        transforms.Normalize([0.485, 0.45
                                       ])
#         print(Image.open(self.X[index]))
        image = transform(torch.tensor(np.moveaxis(np.array(Image.open(se
        label = self.y[index].astype(float)

        return image, label

class MyData_test(Dataset):
    def __init__(self, X, y):
        self.X = X
        self.classes = np.unique(y)
        self.y = np.array(y)
        self.tumorid = y.index

    def __len__(self):
        return len(self.X)

    def __getitem__(self, index):
        transform = transforms.Compose([
                                        transforms.Normalize([0.485, 0.45
                                       ])
        image = transform(torch.tensor(np.moveaxis(np.array(Image.open(se
        label = self.y[index].astype(float)

        return image, label
```

In [18]:
```
eeres_threshold = brca_earlyes_es["HALLMARK_ESTROGEN_RESPONSE_EARLY"].qua
print("EERES threshold:", eeres_threshold)
y_eeres = (brca_earlyes_es>eeres_threshold).applymap(lambda x: 1 if x els
meta_y = y_eeres.join(meta_nondup, how='inner')

files = ("Images/" + meta_y["file_id"] + '/' + meta_y["file_name"] + "-1.

X_train, X_test, y_train, y_test = train_test_split(files, meta_y['HALLMA
print("Total number of data:", len(files))
# 5-fold CV with X_train/y_train
train_dataset = MyData_test(X_train, y_train)
train_dataloader = DataLoader(train_dataset,batch_size=1,shuffle=False)

cv_files = {}
dataset_sizes = {}
kf = KFold(n_splits=5, random_state=0, shuffle=True)
for i, (train_index, val_index) in enumerate(kf.split(X_train)):
    # Double training dataset with cropped and augmented images
    print(f"Fold {i} train data: {len(train_index)}, val data: {len(val_i
    train_file_2 = [file.replace('-1.jpg', '-2.jpg') for file in files[tr
    X_train_double = np.concatenate([files[train_index], train_file_2])
    y_fold_train = meta_y['HALLMARK_ESTROGEN_RESPONSE_EARLY'].iloc[train_
    y_train_double = pd.concat([y_fold_train, y_fold_train])
    X_val = files[val_index]
    y_val = meta_y['HALLMARK_ESTROGEN_RESPONSE_EARLY'].iloc[val_index]
    train_dataset = MyData_train(X_train_double, y_train_double)
    val_dataset = MyData_test(X_val, y_val)
    cv_files[i] = {'train': DataLoader(train_dataset,batch_size=2,shuffle
    dataset_sizes[i] = {'train': len(train_dataset), 'val': len(val_datas
# Testing data with X_test/y_test
print(f"Test data: {len(X_test)}")
test_dataset = MyData_test(X_test, y_test)
test_dataloader = DataLoader(test_dataset,batch_size=1,shuffle=False)
```

```
EERES threshold: 0.0225090214649534
Total number of data: 1046
Fold 0 train data: 627, val data: 157
Fold 1 train data: 627, val data: 157
Fold 2 train data: 627, val data: 157
Fold 3 train data: 627, val data: 157
Fold 4 train data: 628, val data: 156
Test data: 262
```

Training

In [ ]:
```
folder = "Output"
!mkdir "Output"
```

In [20]:
```
# Training function

def train_model(model, criterion, optimizer, scheduler, num_epochs=25, fo
    device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu
    since = time.time()
    checkpoint = torch.load(folder+"/model_initial.pt", map_location=torc
```

```
    for fold in folds:
        best_auroc = 0.0
        best_average_loss = 100
        model.load_state_dict(checkpoint['model_state_dict'], strict=Fals
        model.to(device)
        optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
        folder_fold = folder+"/"+str(fold)
        !mkdir "{folder_fold}"
        for epoch in range(num_epochs):
            print(f'Epoch {epoch}/{num_epochs - 1}')
            print('-' * 10)

            # Each epoch has a training and validation phase
            train_epoch_loss = 100
            val_epoch_loss = 100
            for phase in ['train', 'val']:
                if phase == 'train':
                    model.train()  # Set model to training mode
                else:
                    model.eval()   # Set model to evaluate mode

                running_loss = 0.0
                running_corrects = 0
                running_scores = []
                running_golds = []
                # Iterate over data.
                for inputs, labels in cv_files[fold][phase]:
                    inputs = inputs.to(device)
                    labels = labels.reshape(-1,1).to(device)

                    # zero the parameter gradients
                    optimizer.zero_grad()

                    # forward
                    # track history if only in train
                    with torch.set_grad_enabled(phase == 'train'):
                        outputs = model(inputs)
                        running_scores += torch.flatten(outputs[:,1]).tol
                        running_golds += torch.flatten(labels).tolist()
                        _, preds = torch.max(outputs, 1)
                        preds = torch.reshape(preds, (-1,1)).float()
                        gold = torch.tensor([[1,0] if label==0 else [0,1]
                        loss = criterion(outputs.float(), gold.float())

                        # backward + optimize only if in training phase
                        if phase == 'train':
                            loss.backward()
                            optimizer.step()

                    # statistics
                    running_loss += loss.item() * inputs.size(0)
                    running_corrects += torch.sum(preds == labels.data)
                if phase == 'train':
                    scheduler.step()

                epoch_loss = running_loss / dataset_sizes[fold][phase]
                epoch_auroc = roc_auc_score(running_golds, running_scores
```

```
                if phase == 'train':
                    train_epoch_loss = epoch_loss
                elif phase == 'val':
                    val_epoch_loss = epoch_loss
                    average_loss = (train_epoch_loss+val_epoch_loss)/2
                    print(f'{phase} Average Loss: {average_loss}')

                print(f'{phase} Loss: {epoch_loss:.4f} AUROC {epoch_auroc

                # Save the model with highest val AUROC
                if phase == 'val' and best_auroc < epoch_auroc:
                    best_average_loss = average_loss
                    best_auroc = epoch_auroc
                    torch.save({
                    'epoch': epoch,
                    'model_state_dict': model.state_dict(),
                    'optimizer_state_dict': optimizer.state_dict(),
                    'loss': loss,
                    }, folder_fold+"/epoch_"+str(epoch)+"_auroc_"+str(epo

            print()

        time_elapsed = time.time() - since
        print(f'Training complete in {time_elapsed // 60:.0f}m {time_elap
        print(f'Best val AUROC: {best_auroc:4f}')
```

In [21]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Transfer learning with ResNet50 and followed by 3 fc layers

model_ft = models.resnet50(weights='DEFAULT').to(device)
import torch.nn.functional as F
class net(nn.Module):
    def __init__(self):
        super(net, self).__init__()
        self.fc1 = nn.Linear(1000, 128)
        self.fc2 = nn.Linear(128, 32)
        self.fc3 = nn.Linear(32, 2)
        self.m = nn.Dropout(p=0.2)

    def forward(self, x):
        x = self.m(x)
        x = F.relu(self.fc1(x))
        x = self.m(x)
        x = F.relu(self.fc2(x))
        x = self.m(x)
        x = F.softmax(self.fc3(x), dim=1)
        return x

net_add = net()

model_ft = nn.Sequential(model_ft, net_add).to(device)

# Observe that all parameters are being optimized
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.01, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0

# Save initial model parameters for each fold of CV
torch.save({
            'model_state_dict': model_ft.state_dict(),
            'optimizer_state_dict': optimizer_ft.state_dict(),
            }, folder+"/model_initial.pt")

criterion = nn.CrossEntropyLoss()
```

In [ ]:
```
train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
                       num_epochs=25, folder=folder, folds=[0,1,2,3,4])
```

Evaluation with unseen testing data

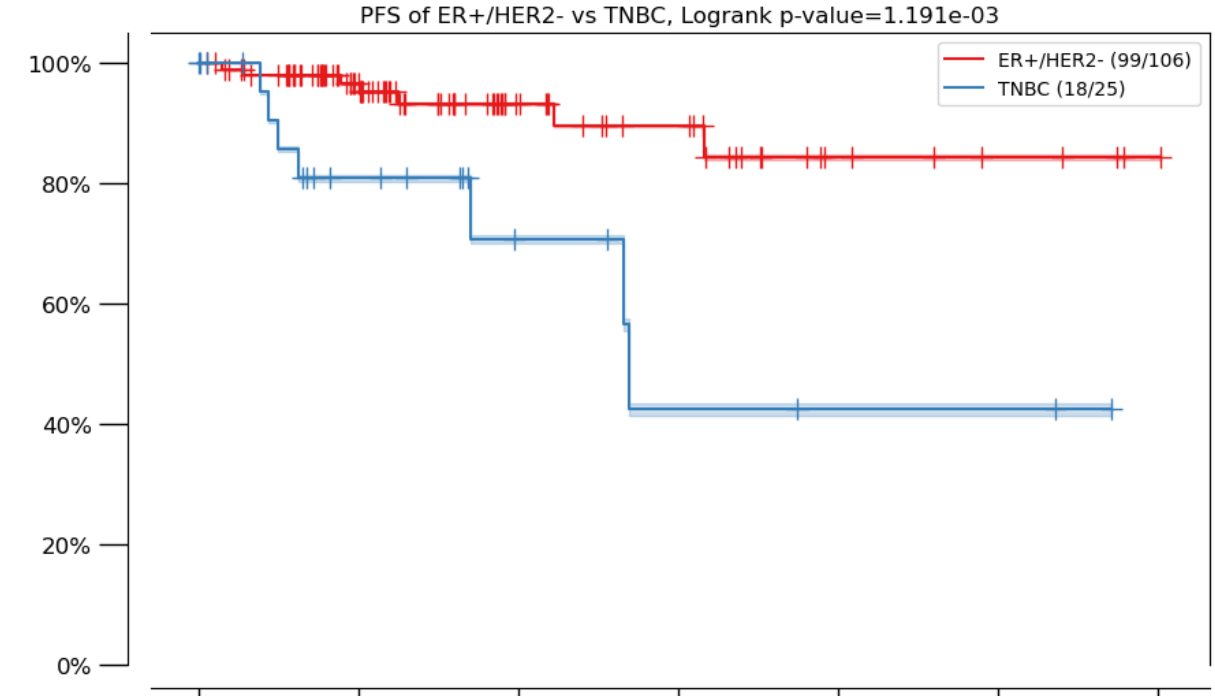
In [22]:
```
cv_test_index = y_test.index
cv_predicted_df_dict = {}

for fold in range(5):
    cv_test_predicted_df = pd.DataFrame(np.zeros((len(cv_test_index),2)),
    file = np.sort(os.listdir(f"{folder}/{str(fold)}"))[-1] # Get the las
    print(f"Fold {fold} model:", file)
    model_path = f"{folder}/{str(fold)}/{file}"
    checkpoint = torch.load(model_path, map_location=torch.device('cpu'))
    model_ft = models.resnet50().to(device)
    model_test = nn.Sequential(model_ft, net_add).to(device)
    model_test.load_state_dict(checkpoint['model_state_dict'])
    model_test.cuda()
    criterion = nn.CrossEntropyLoss()
    epoch_test = checkpoint['epoch']
    loss_test = checkpoint['loss']
    model_test.eval()

    loss_epoch_test=[]
    y_proba = []
    y_gold = []
    y_pred = []
    with torch.no_grad():
        for b, (X, y) in enumerate(test_dataloader):
            outputs = model_test(X.cuda())
            _, preds = torch.max(outputs, 1)
            y_proba += torch.flatten(outputs[:,1]).cpu().tolist()
            y_gold += y.data.cpu().tolist()
            y_pred += preds.cpu().tolist()
#             loss = criterion(outputs.float(), torch.tensor([[1,0] if la
#             loss_epoch_test.append(loss.item())

        auc=roc_auc_score(y_gold,y_proba)
        print(f"Fold {fold} AUROC: {auc}")
    cv_test_predicted_df.loc[cv_test_index,'score'] = y_proba
    cv_test_predicted_df.loc[cv_test_index,'gold'] = y_gold
    cv_predicted_df_dict[fold] = cv_test_predicted_df

# Average the scores of the 5 folds
cv_test_predicted_df["score"] = (cv_predicted_df_dict[0]["score"]+\
                            cv_predicted_df_dict[1]["score"]+\
                            cv_predicted_df_dict[2]["score"]+\
                            cv_predicted_df_dict[3]["score"]+\
                            cv_predicted_df_dict[4]["score"])/5
```

```
Fold 0 model: epoch_21_loss_0.617512_auroc_0.729207.pt
Fold 0 AUROC: 0.7536934306569344
Fold 1 model: epoch_9_loss_0.674120_auroc_0.596405.pt
Fold 1 AUROC: 0.7063941605839417
Fold 2 model: epoch_6_loss_0.676420_auroc_0.628654.pt
Fold 2 AUROC: 0.6602043795620438
Fold 3 model: epoch_9_loss_0.650217_auroc_0.667819.pt
Fold 3 AUROC: 0.728992700729927
Fold 4 model: epoch_8_loss_0.667883_auroc_0.647039.pt
Fold 4 AUROC: 0.7028321167883211
```

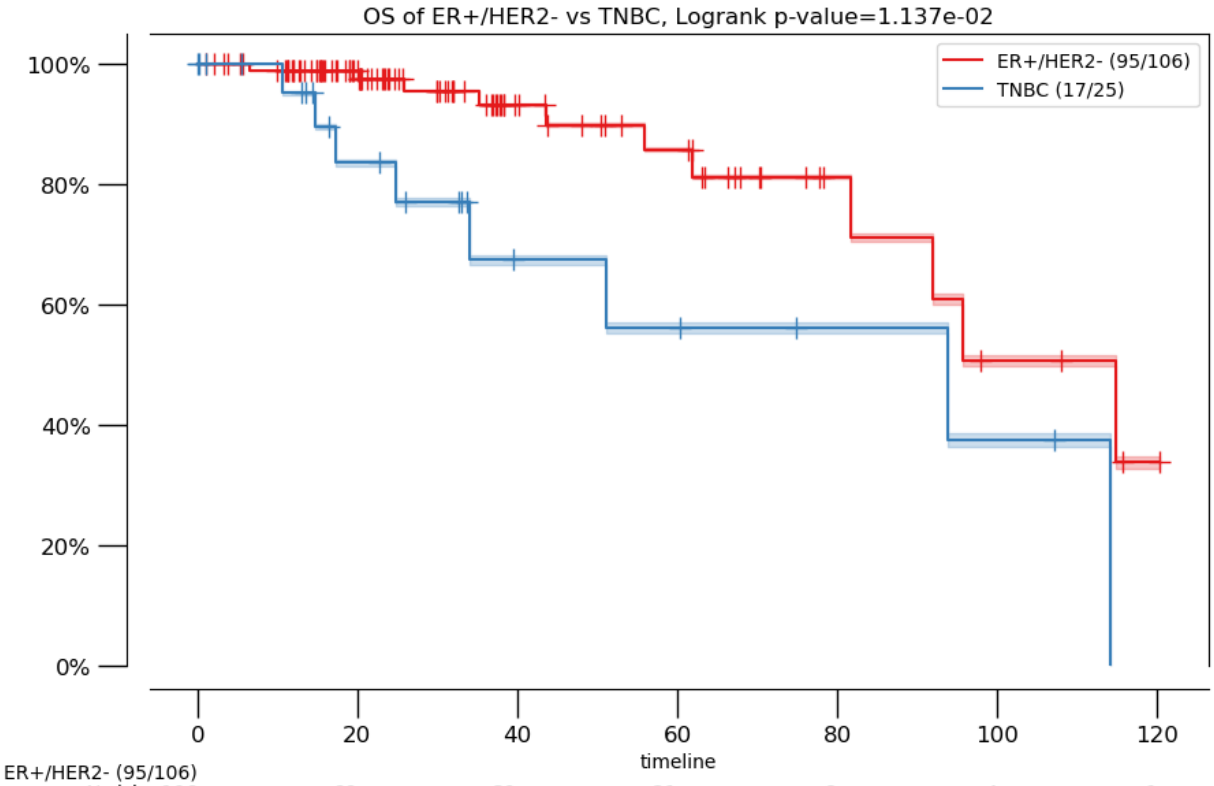
Examine the Survival of the testing data (ER+/HER2- vs TNBC)

In [23]:
```
cv_test_predicted_df_ihc = cv_test_predicted_df.join(meta_erihc, how='inn
cv_test_predicted_df_ihc["Subtype"] = None
cv_test_predicted_df_ihc["Subtype"][(cv_test_predicted_df_ihc["er_status_
# cv_predicted_df_ihc["Subtype"][cv_predicted_df_ihc["her2_status_by_ihc"
cv_test_predicted_df_ihc["Subtype"][(cv_test_predicted_df_ihc["er_status_
cv_test_predicted_df_ihc["EERES"] = brca_earlyes_es.loc[cv_test_predicted
cv_test_predicted_df_ihc["ESR1"] = brca_lv3.loc[cv_test_predicted_df_ihc.

brca_clinical_testing = brca_clinical.join(cv_test_predicted_df_ihc, how=
brca_clinical_testing_dfs = brca_clinical_testing[["DFS_MONTHS", "DFS_STA
brca_clinical_testing_dss = brca_clinical_testing[["DSS_MONTHS", "DSS_STA
brca_clinical_testing_pfs = brca_clinical_testing[["PFS_MONTHS", "PFS_STA
brca_clinical_testing_os = brca_clinical_testing[["OS_MONTHS", "OS_STATUS

result = km.fit(brca_clinical_testing_pfs['PFS_MONTHS'], brca_clinical_te
km.plot(result, title=f"PFS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_testing_os['OS_MONTHS'], brca_clinical_test
km.plot(result, title=f"OS of ER+/HER2- vs TNBC, Logrank p-value={result[
plt.show()
result = km.fit(brca_clinical_testing_dfs['DFS_MONTHS'], brca_clinical_te
km.plot(result, title=f"DFS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_testing_dss['DSS_MONTHS'], brca_clinical_te
km.plot(result, title=f"DSS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()


plt.scatter(cv_test_predicted_df_ihc[cv_test_predicted_df_ihc["Subtype"]=
# plt.scatter(cv_predicted_df_ihc[cv_predicted_df_ihc["Subtype"]=="ERBB2+
plt.scatter(cv_test_predicted_df_ihc[cv_test_predicted_df_ihc["Subtype"]=
plt.legend(["ER+/HER2-", "TNBC"])
plt.xlabel("ESR1")
plt.ylabel("EERES")
plt.show()
```

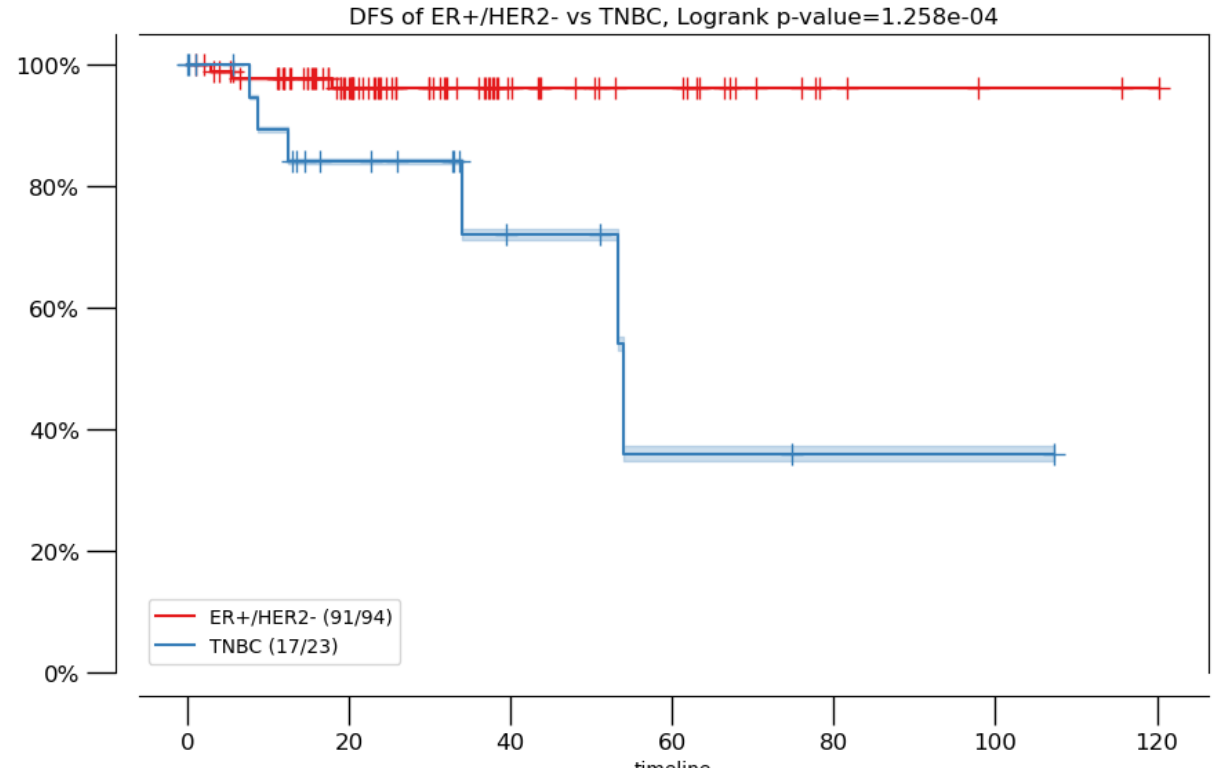
```
/tmp/ipykernel_93586/4221478194.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cv_test_predicted_df_ihc["Subtype"][(cv_test_predicted_df_ihc["er_stat
us_by_ihc"]=='Positive') & (cv_test_predicted_df_ihc["her2_status_by_ihc
"]=='Negative')] = "ER+/HER2-"
/tmp/ipykernel_93586/4221478194.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cv_test_predicted_df_ihc["Subtype"][(cv_test_predicted_df_ihc["er_stat
us_by_ihc"]=='Negative') & (cv_test_predicted_df_ihc["her2_status_by_ihc
"]=='Negative') & (cv_test_predicted_df_ihc["pr_status_by_ihc"]=='Negati
ve')] = "TNBC"
```



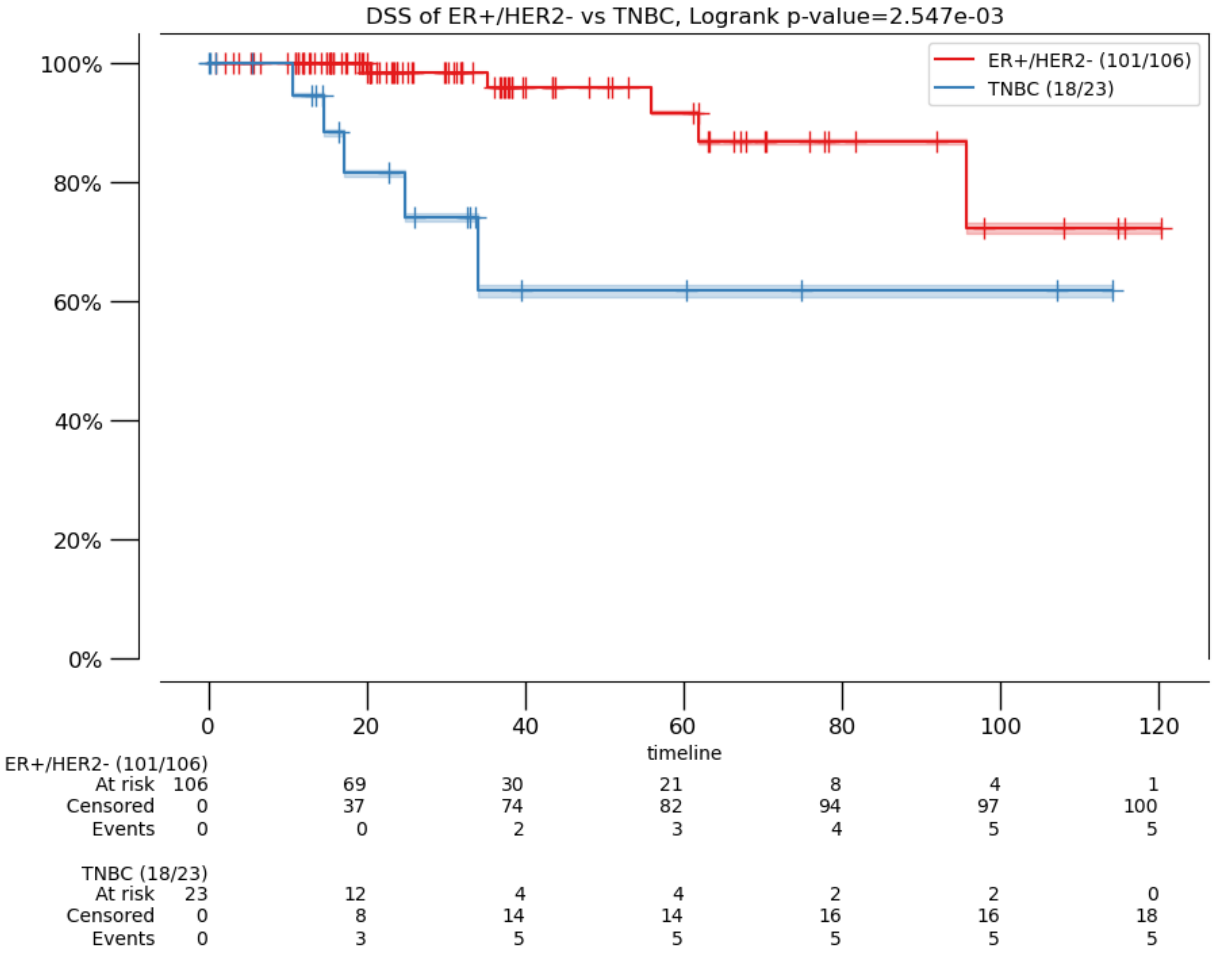
ER+/HER2- (99/106)							
At risk	106	67	29	20	7	4	1
Censored	0	36	72	80	92	95	98
Events	0	3	5	6	7	7	7
TNBC (18/25)							
At risk	25	13	6	3	2	2	0
Censored	0	8	14	15	16	16	18
Events	0	4	5	7	7	7	7

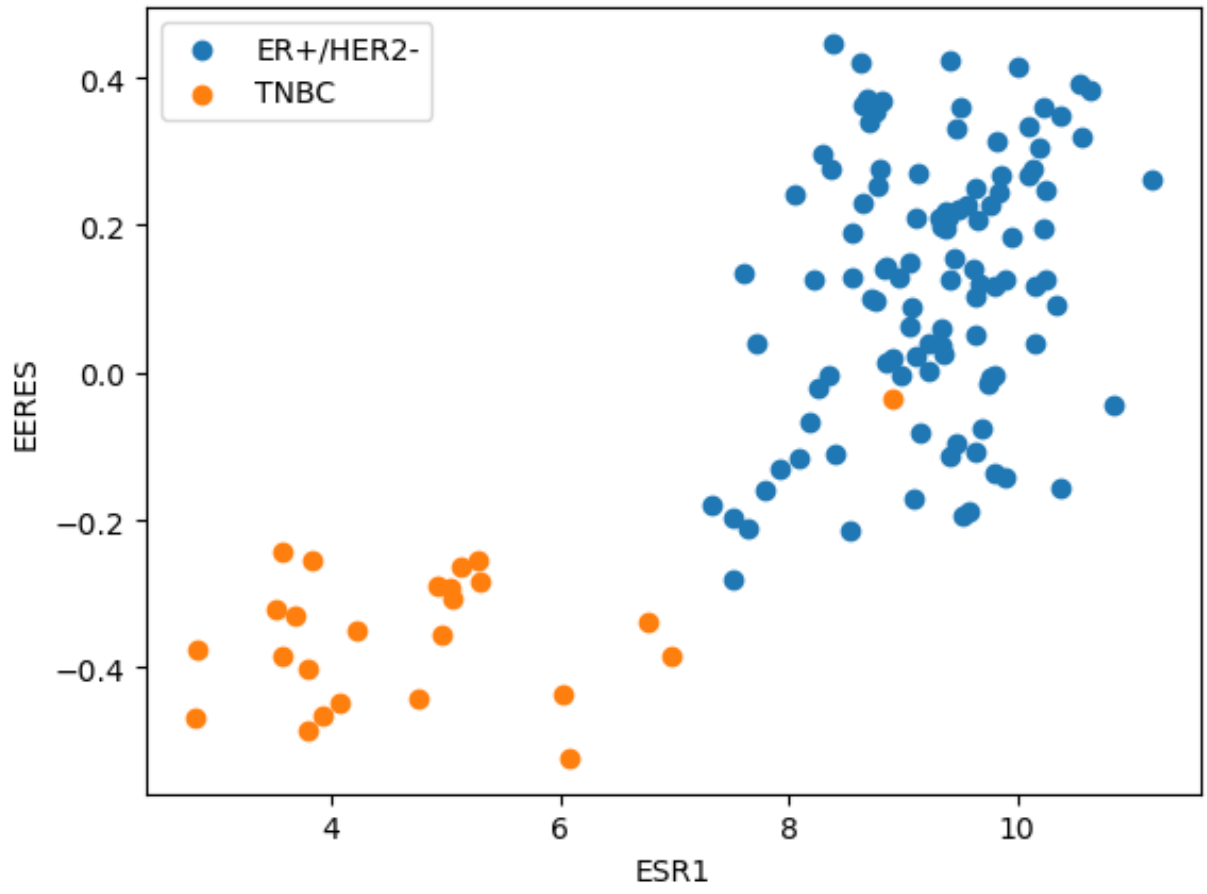


ER+/HER2- (95/106)							
At risk	106	69	30	21	8	4	1
Censored	0	36	72	79	91	92	94
Events	0	1	4	6	7	10	11
TNBC (17/25)							
At risk	25	14	6	5	3	2	0
Censored	0	8	14	14	16	16	17
Events	0	3	5	6	6	7	8



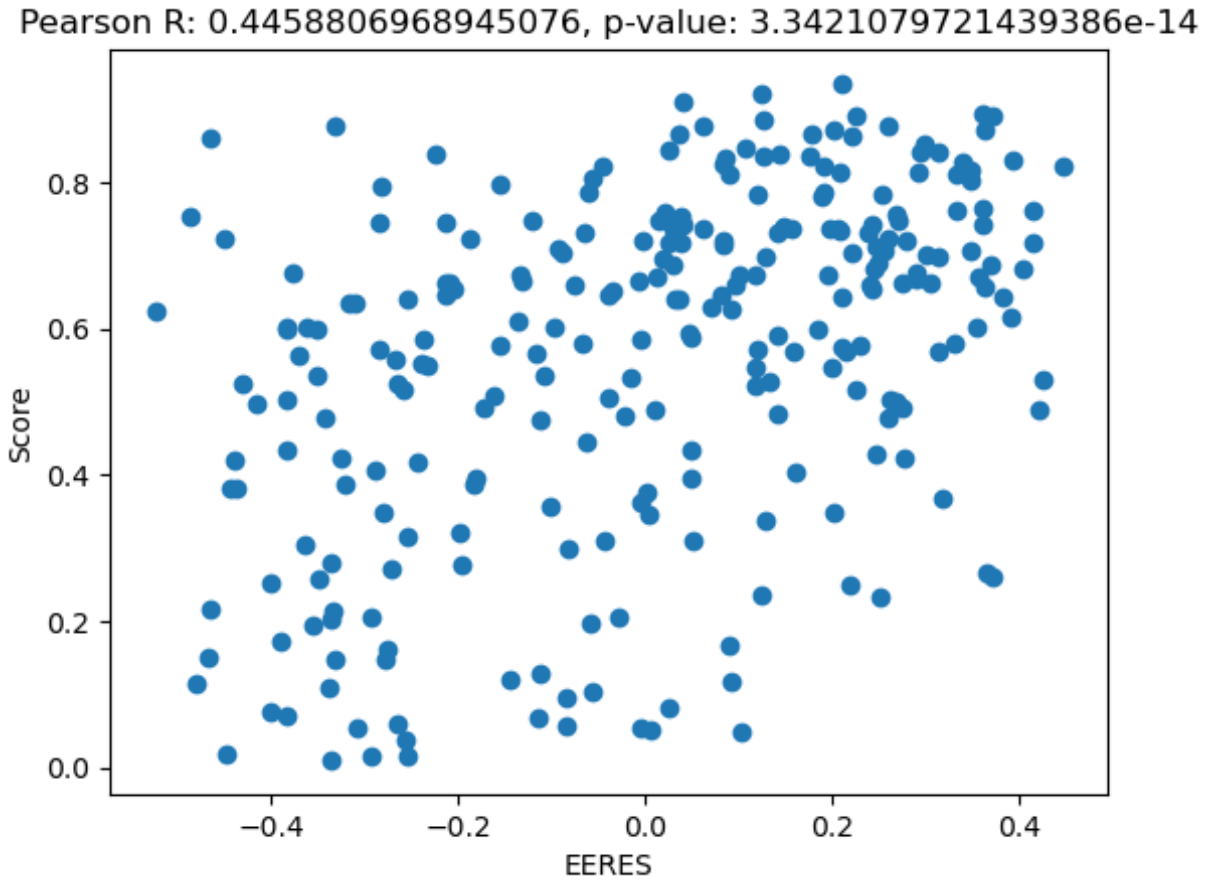
ER+/HER2- (91/94)							
At risk	94	59	23	15	4	2	1
Censored	0	32	68	76	87	89	90
Events	0	3	3	3	3	3	3
TNBC (17/23)							
At risk	23	12	5	2	1	1	0
Censored	0	8	14	15	16	16	17
Events	0	3	4	6	6	6	6

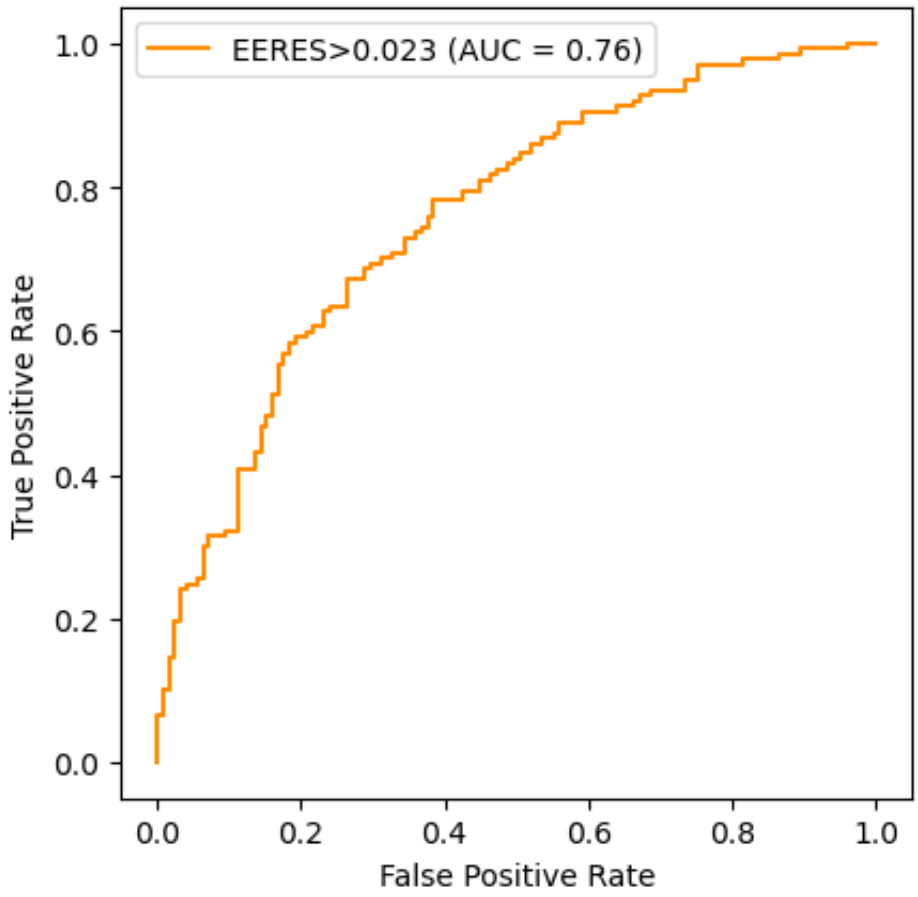




In [24]:
```
cv_test_predicted_df_ihc[['score','EERES','her2_status_by_ihc','er_status
```

In [33]:
```
plt.scatter(cv_test_predicted_df_ihc["EERES"], cv_test_predicted_df_ihc["
plt.xlabel("EERES")
plt.ylabel("Score")
r, p = pearsonr(cv_test_predicted_df_ihc["EERES"], cv_test_predicted_df_i
plt.title(f"Pearson R: {r}, p-value: {p}")

auroc = roc_auc_score(cv_test_predicted_df_ihc["gold"], cv_test_predicted
RocCurveDisplay.from_predictions(
    cv_test_predicted_df_ihc["gold"],
    cv_test_predicted_df_ihc["score"],
    name=f"EERES>{eeres_threshold:.3f}",
    color="darkorange",
)
# plt.plot([0, 1], [0, 1], "k--", label="chance level (AUC = 0.5)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
# plt.title("Receiver Operating Characteristic")
plt.legend()
plt.show()
```

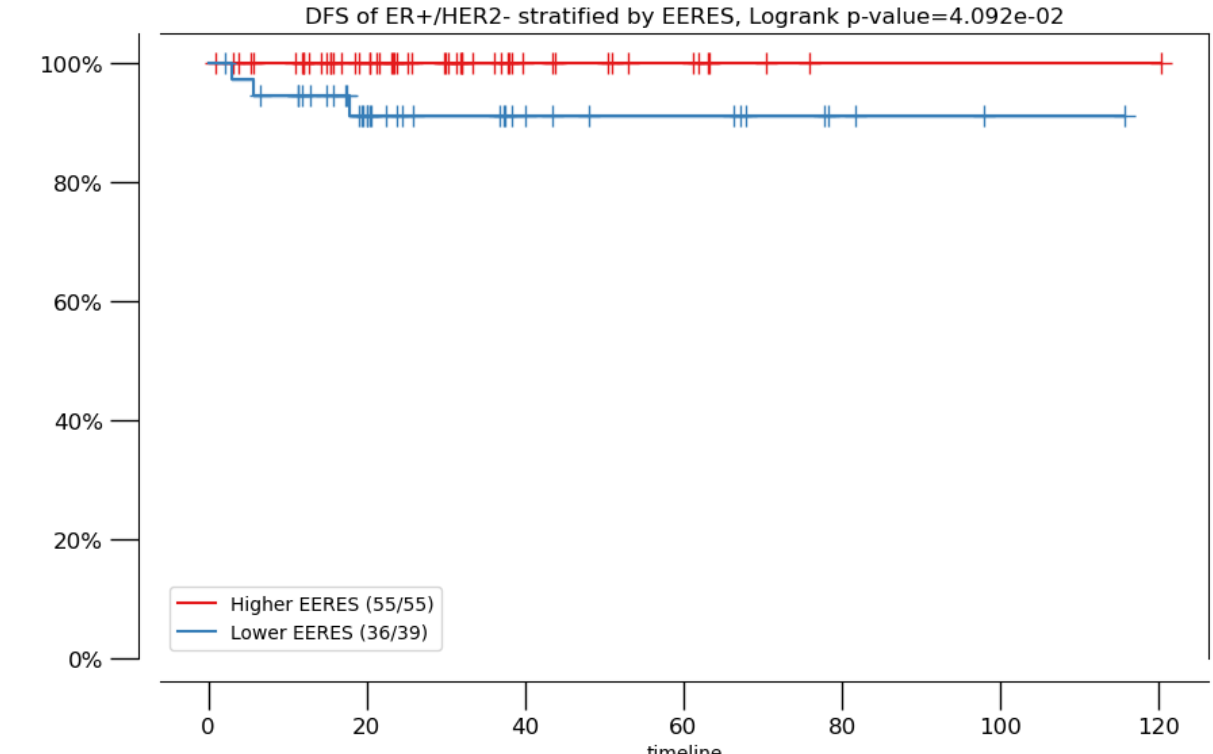




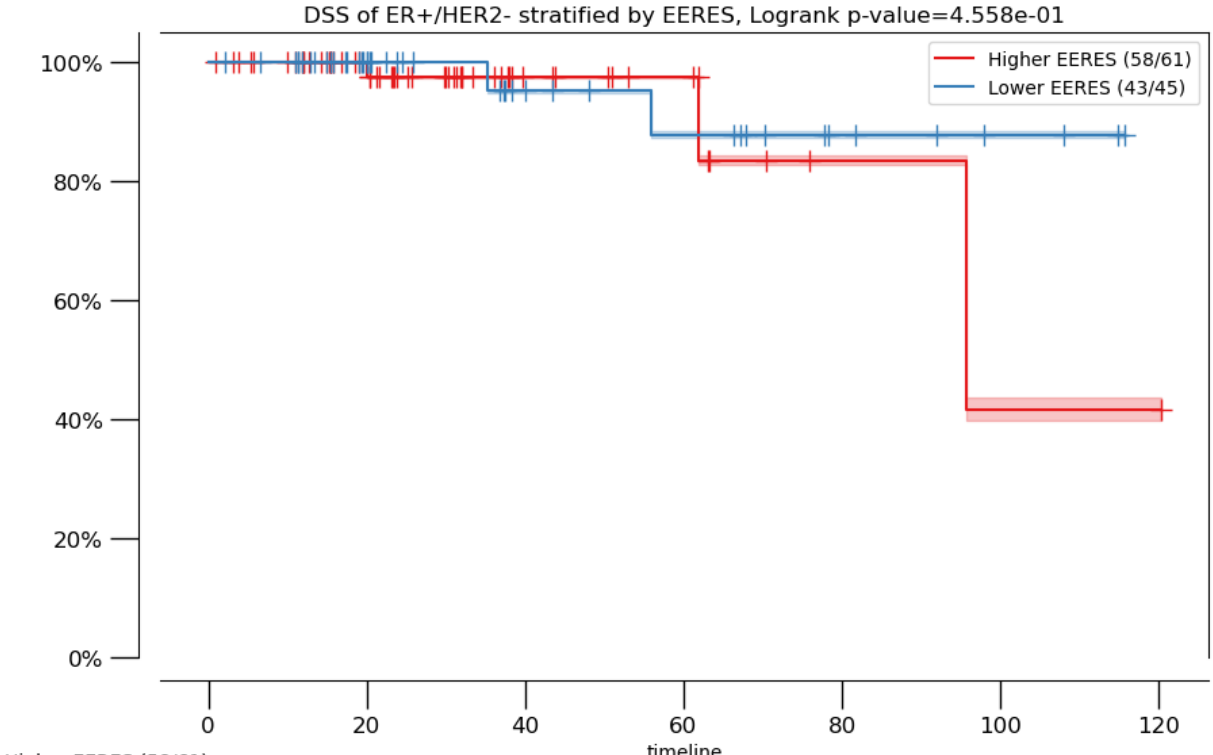
Examine the Survival of the ER+/HER2- testing data (Higher/Lower Predicted Score)

In [90]:
```
brca_clinical_testing_erposerbb2neg = brca_clinical.join(cv_test_predicte
brca_clinical_testing_erposerbb2neg

brca_clinical_testing_erposerbb2neg_dfs = brca_clinical_testing_erposerbb
brca_clinical_testing_erposerbb2neg_dss = brca_clinical_testing_erposerbb
brca_clinical_testing_erposerbb2neg_pfs = brca_clinical_testing_erposerbb
brca_clinical_testing_erposerbb2neg_os = brca_clinical_testing_erposerbb2
finding_best_threshold_with_FS_SS(brca_clinical_testing_erposerbb2neg_dfs
finding_best_threshold_with_FS_SS(brca_clinical_testing_erposerbb2neg_pfs
finding_best_threshold_with_FS_SS(brca_clinical_testing_erposerbb2neg_dfs
finding_best_threshold_with_FS_SS(brca_clinical_testing_erposerbb2neg_pfs
```

```
Best EERES threshold: 0.1109585485096507
```

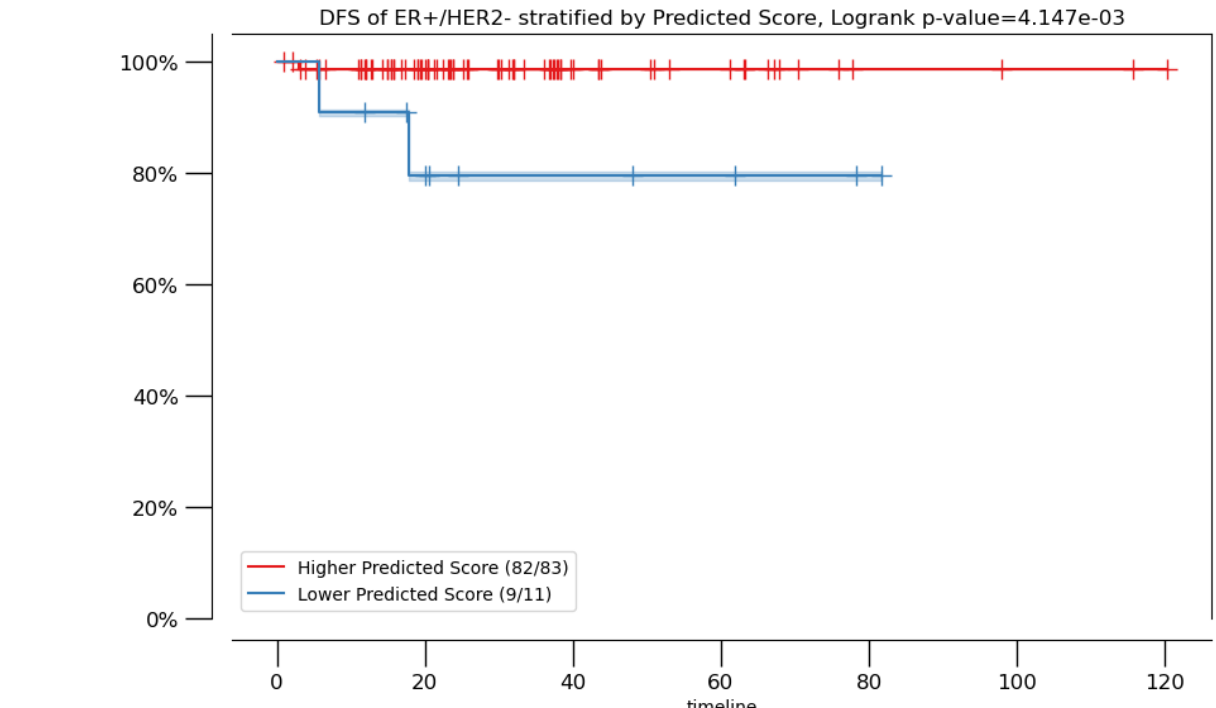


Higher EERES (55/55)							
At risk	55	36	12	7	1	1	1
Censored	0	19	43	48	54	54	54
Events	0	0	0	0	0	0	0
Lower EERES (36/39)							
At risk	39	23	11	8	3	1	0
Censored	0	13	25	28	33	35	36
Events	0	3	3	3	3	3	3

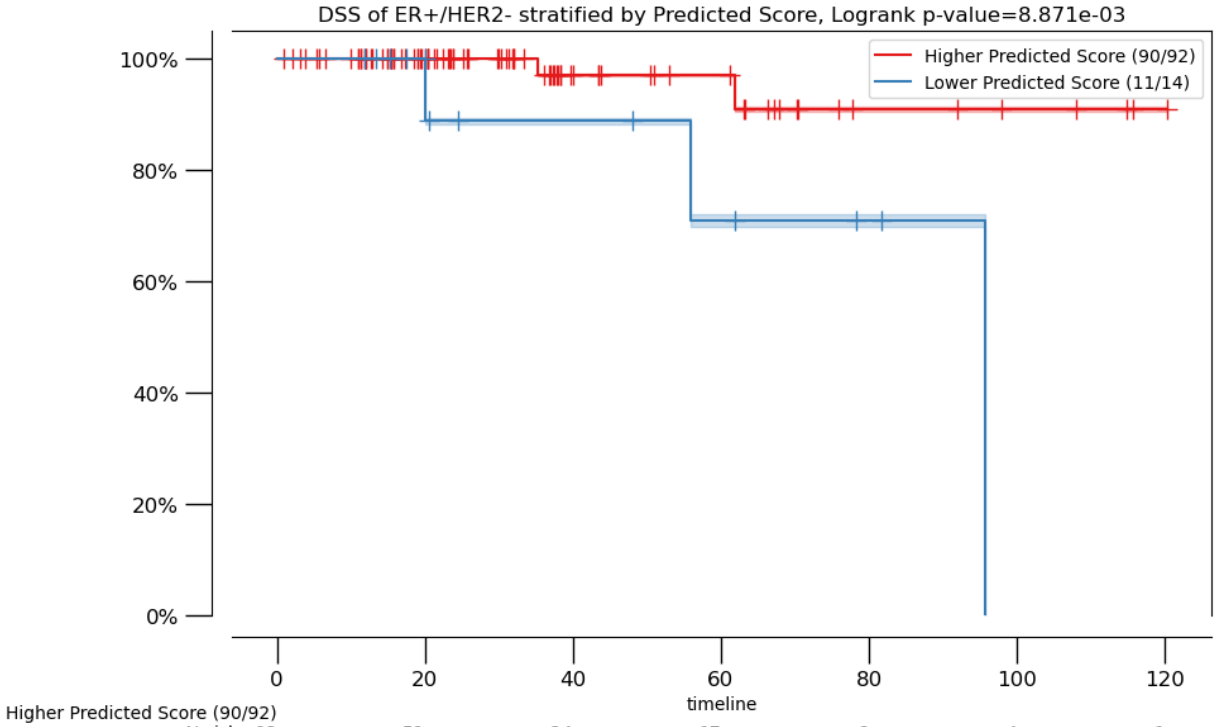


Higher EERES (58/61)							
At risk	61	40	14	9	2	1	1
Censored	0	21	46	51	57	57	57
Events	0	0	1	1	2	3	3
Lower EERES (43/45)							
At risk	45	29	16	12	6	3	0
Censored	0	16	28	31	37	40	43
Events	0	0	1	2	2	2	2

not significant
Best score threshold: 0.3645404119975865

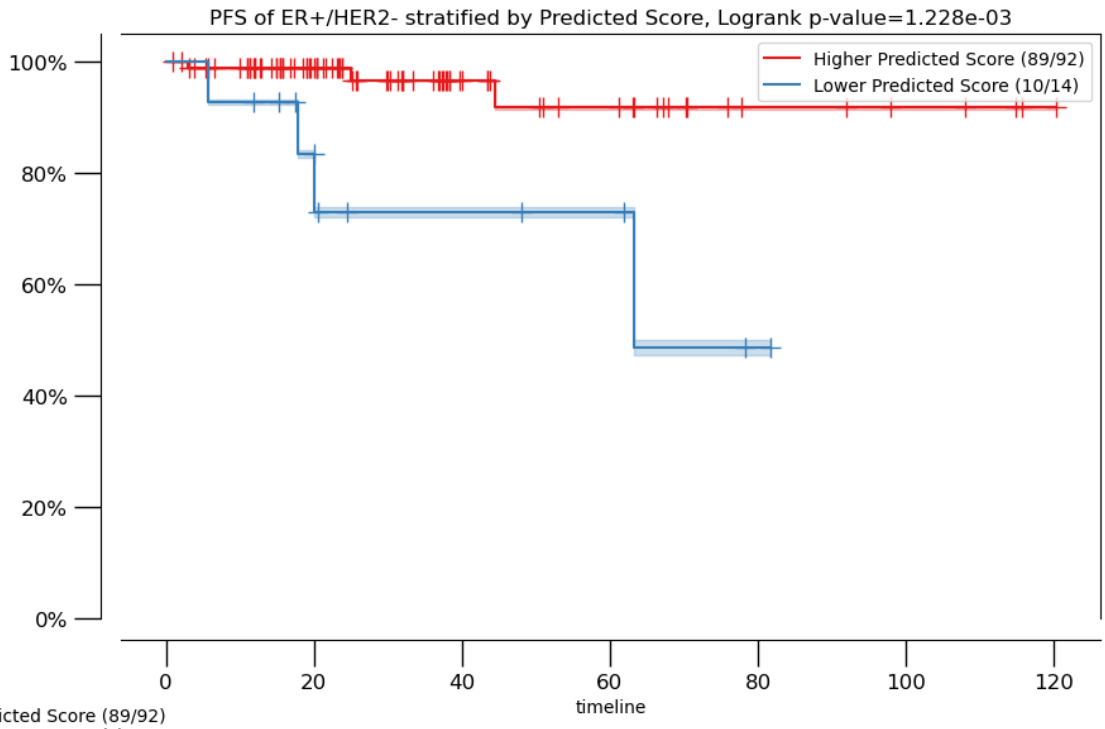


Higher Predicted Score (82/83)							
At risk	83	52	19	12	3	2	1
Censored	0	30	63	70	79	80	81
Events	0	1	1	1	1	1	1
Lower Predicted Score (9/11)							
At risk	11	7	4	3	1	0	0
Censored	0	2	5	6	8	9	9
Events	0	2	2	2	2	2	2

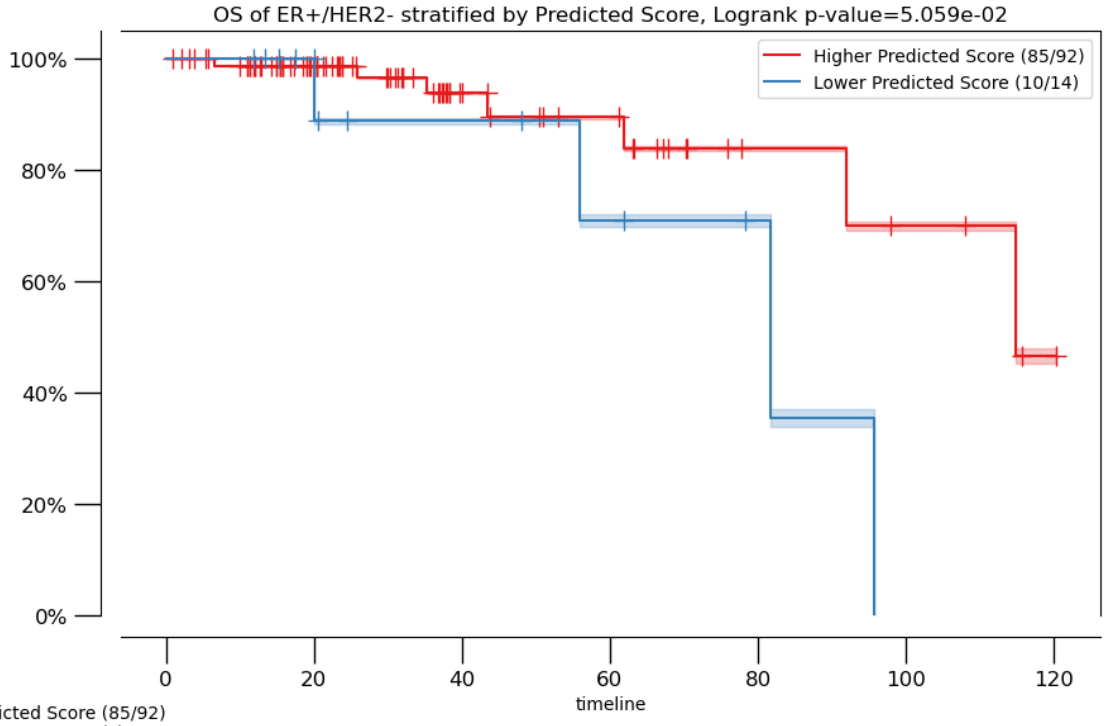


Higher Predicted Score (90/92)							
At risk	92	59	24	17	6	4	1
Censored	0	33	67	74	84	86	89
Events	0	0	1	1	2	2	2
Lower Predicted Score (11/14)							
At risk	14	10	6	4	2	0	0
Censored	0	4	7	8	10	11	11
Events	0	0	1	2	2	3	3

Best score threshold: 0.3645404119975865



Higher Predicted Score (89/92)							
At risk	92	58	24	16	6	4	1
Censored	0	33	66	73	83	85	88
Events	0	1	2	3	3	3	3
Lower Predicted Score (10/14)							
At risk	14	9	5	4	1	0	0
Censored	0	3	6	7	9	10	10
Events	0	2	3	3	4	4	4



Higher Predicted Score (85/92)							
At risk	92	59	24	17	6	4	1
Censored	0	32	65	71	81	82	84
Events	0	1	3	4	5	6	7
Lower Predicted Score (10/14)							
At risk	14	10	6	4	2	0	0
Censored	0	4	7	8	10	10	10
Events	0	0	1	2	2	4	4

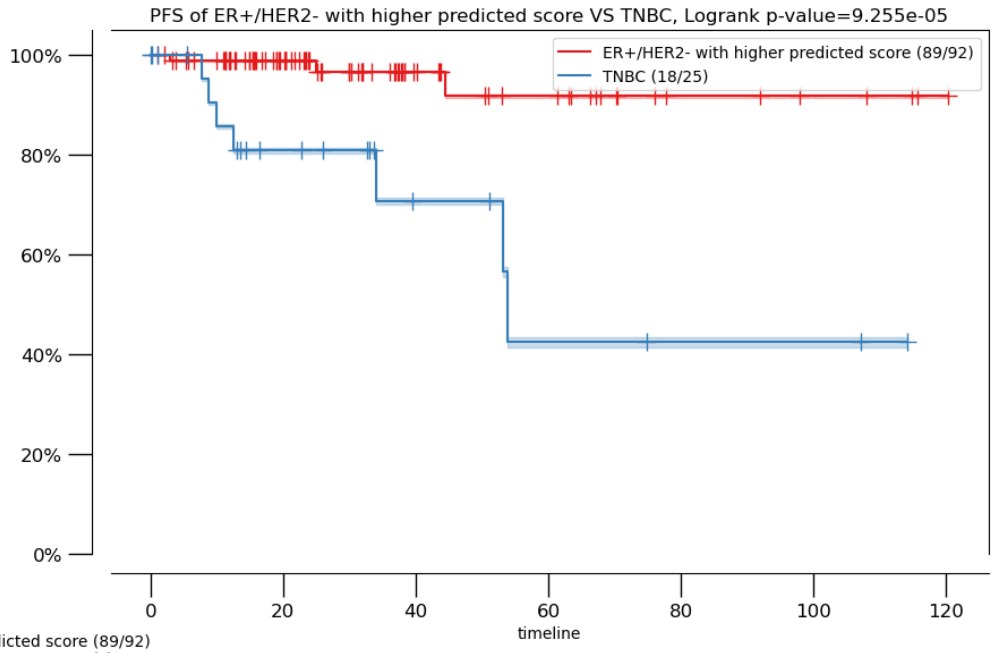
In [47]:
```
brca_clinical_tnbc = brca_clinical.join(cv_test_predicted_df_ihc[cv_test_

brca_clinical_tnbc_dfs = brca_clinical_tnbc[["DFS_MONTHS", "DFS_STATUS",
brca_clinical_tnbc_dss = brca_clinical_tnbc[["DSS_MONTHS", "DSS_STATUS",
brca_clinical_tnbc_pfs = brca_clinical_tnbc[["PFS_MONTHS", "PFS_STATUS",
brca_clinical_tnbc_os = brca_clinical_tnbc[["OS_MONTHS", "OS_STATUS", "sc

finding_best_threshold_with_FS_SS(brca_clinical_tnbc_dfs, brca_clinical_t
finding_best_threshold_with_FS_SS(brca_clinical_tnbc_dfs, brca_clinical_t
```

```
not significant
not significant
```

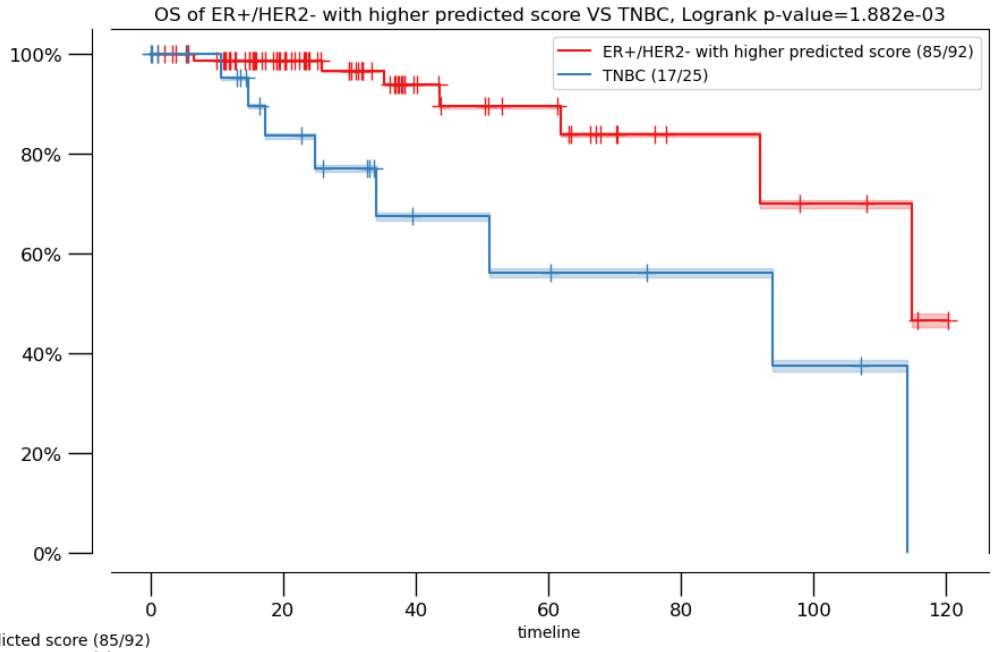
Higher/Lower Predicted Score vs TNBC Survival

In [105…

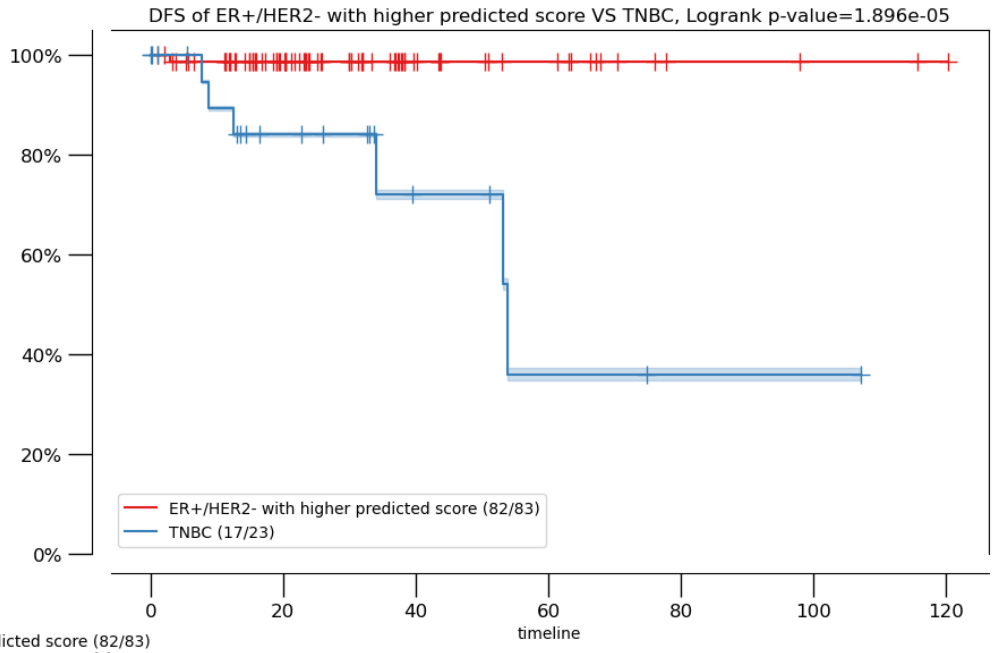
```
erposerbb2neg_highscore_tnbc_df = pd.concat([cv_test_predicted_df_ihc[cv_
cv_test_predicted_df_ihc[(cv_test_predicted_df_ihc['Subtype']=="ER+/HER2-
erposerbb2neg_highscore_tnbc_df.loc[erposerbb2neg_highscore_tnbc_df["Subt
brca_clinical_erposerbb2neg_highscore_tnbc = brca_clinical.join(erposerbb

erposerbb2neg_lowscore_tnbc_df = pd.concat([cv_test_predicted_df_ihc[cv_t
cv_test_predicted_df_ihc[(cv_test_predicted_df_ihc['Subtype']=="ER+/HER2-
erposerbb2neg_lowscore_tnbc_df.loc[erposerbb2neg_lowscore_tnbc_df["Subtyp
brca_clinical_erposerbb2neg_lowscore_tnbc = brca_clinical.join(erposerbb2

erposerbb2neg_highscore_tnbc_df_pfs = brca_clinical_erposerbb2neg_highsco
result = km.fit(erposerbb2neg_highscore_tnbc_df_pfs[f'PFS_MONTHS'], erpos
km.plot(result, title=f"PFS of ER+/HER2- with higher predicted score VS T
plt.show()

erposerbb2neg_highscore_tnbc_df_os = brca_clinical_erposerbb2neg_highscor
result = km.fit(erposerbb2neg_highscore_tnbc_df_os[f'OS_MONTHS'], erposer
km.plot(result, title=f"OS of ER+/HER2- with higher predicted score VS TN
plt.show()

erposerbb2neg_highscore_tnbc_df_dfs = brca_clinical_erposerbb2neg_highsco
result = km.fit(erposerbb2neg_highscore_tnbc_df_dfs[f'DFS_MONTHS'], erpos
km.plot(result, title=f"DFS of ER+/HER2- with higher predicted score VS T
plt.show()

erposerbb2neg_highscore_tnbc_df_dss = brca_clinical_erposerbb2neg_highsco
result = km.fit(erposerbb2neg_highscore_tnbc_df_dss[f'DSS_MONTHS'], erpos
km.plot(result, title=f"DSS of ER+/HER2- with higher predicted score VS T
plt.show()

erposerbb2neg_lowscore_tnbc_df_pfs = brca_clinical_erposerbb2neg_lowscore
result = km.fit(erposerbb2neg_lowscore_tnbc_df_pfs[f'PFS_MONTHS'], erpose
km.plot(result, title=f"PFS of ER+/HER2- with lower predicted score VS TN
plt.show()

erposerbb2neg_lowscore_tnbc_df_os = brca_clinical_erposerbb2neg_lowscore_
result = km.fit(erposerbb2neg_lowscore_tnbc_df_os[f'OS_MONTHS'], erposerb
km.plot(result, title=f"OS of ER+/HER2- with lower predicted score VS TNB
plt.show()

erposerbb2neg_lowscore_tnbc_df_dfs = brca_clinical_erposerbb2neg_lowscore
result = km.fit(erposerbb2neg_lowscore_tnbc_df_dfs[f'DFS_MONTHS'], erpose
km.plot(result, title=f"DFS of ER+/HER2- with lower predicted score VS TN
plt.show()

erposerbb2neg_lowscore_tnbc_df_dss = brca_clinical_erposerbb2neg_lowscore
result = km.fit(erposerbb2neg_lowscore_tnbc_df_dss[f'DSS_MONTHS'], erpose
km.plot(result, title=f"DSS of ER+/HER2- with lower predicted score VS TN
plt.show()
```



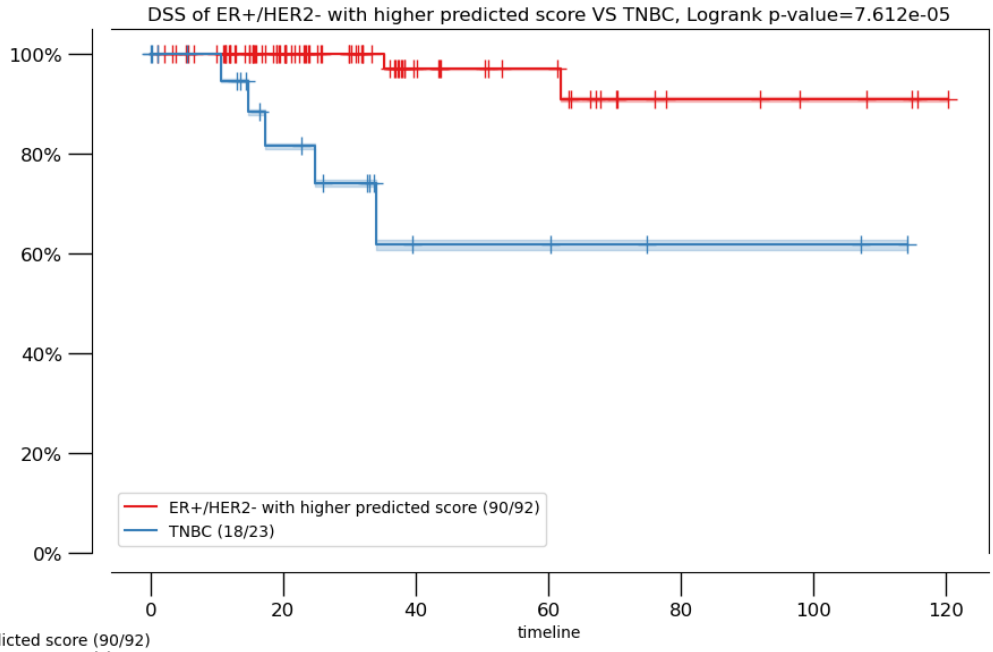
ER+/HER2- with higher predicted score (89/92)							
At risk	92	58	24	16	6	4	1
Censored	0	33	66	73	83	85	88
Events	0	1	2	3	3	3	3
TNBC (18/25)							
At risk	25	13	6	3	2	2	0
Censored	0	8	14	15	16	16	18
Events	0	4	5	7	7	7	7



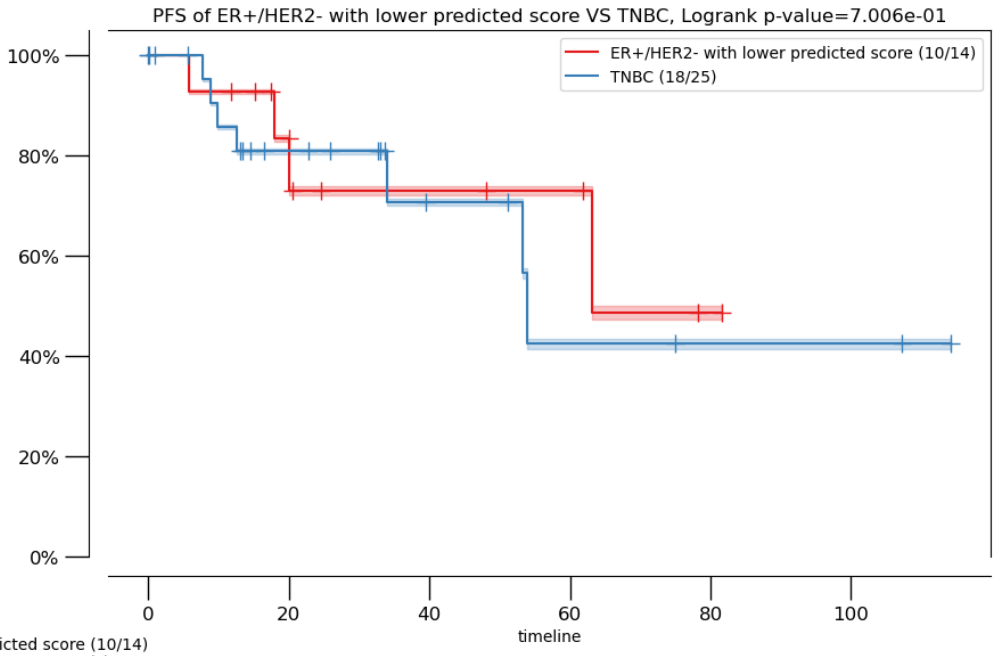
ER+/HER2- with higher predicted score (85/92)							
At risk	92	59	24	17	6	4	1
Censored	0	32	65	71	81	82	84
Events	0	1	3	4	5	6	7
TNBC (17/25)							
At risk	25	14	6	5	3	2	0
Censored	0	8	14	14	16	16	17
Events	0	3	5	6	6	7	8



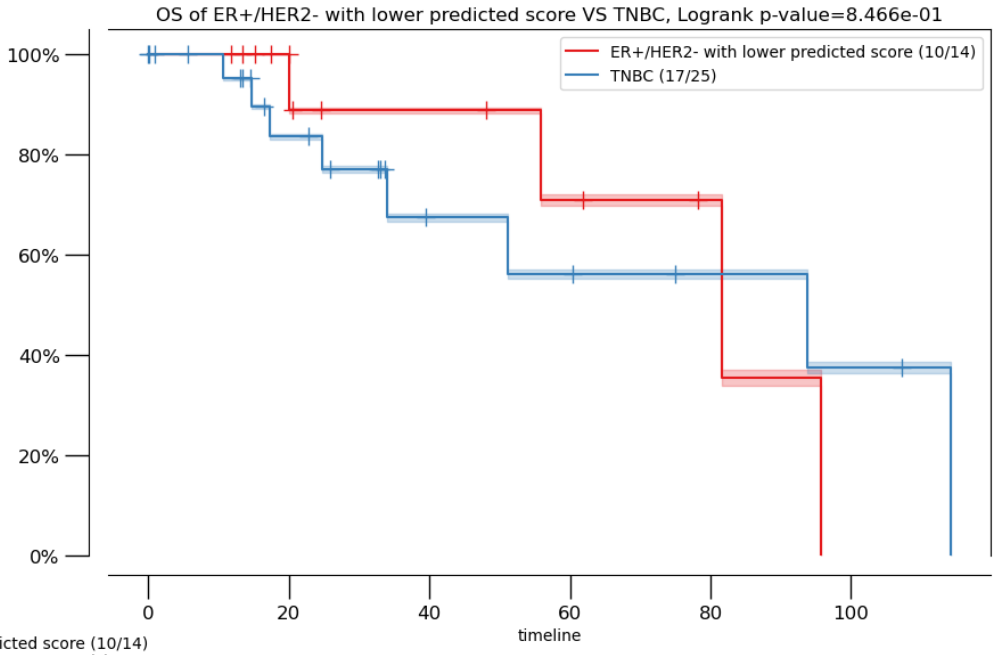
ER+/HER2- with higher predicted score (82/83)							
At risk	83	52	19	12	3	2	1
Censored	0	30	63	70	79	80	81
Events	0	1	1	1	1	1	1
TNBC (17/23)							
At risk	23	12	5	2	1	1	0
Censored	0	8	14	15	16	16	17
Events	0	3	4	6	6	6	6



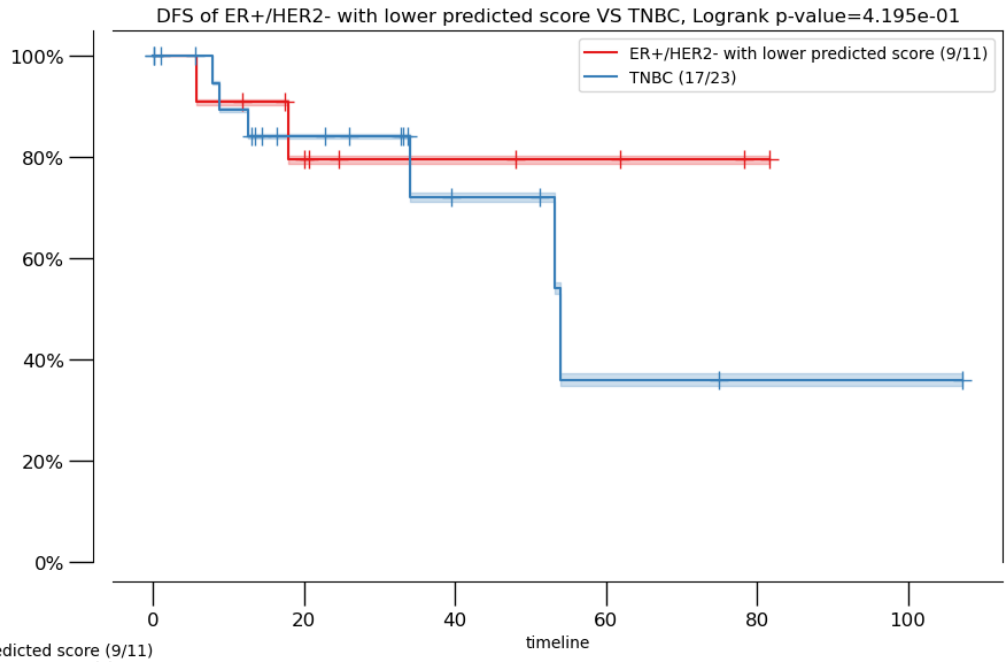
ER+/HER2- with higher predicted score (90/92)							
At risk	92	59	24	17	6	4	1
Censored	0	33	67	74	84	86	89
Events	0	0	1	1	2	2	2
TNBC (18/23)							
At risk	23	12	4	4	2	2	0
Censored	0	8	14	14	16	16	18
Events	0	3	5	5	5	5	5



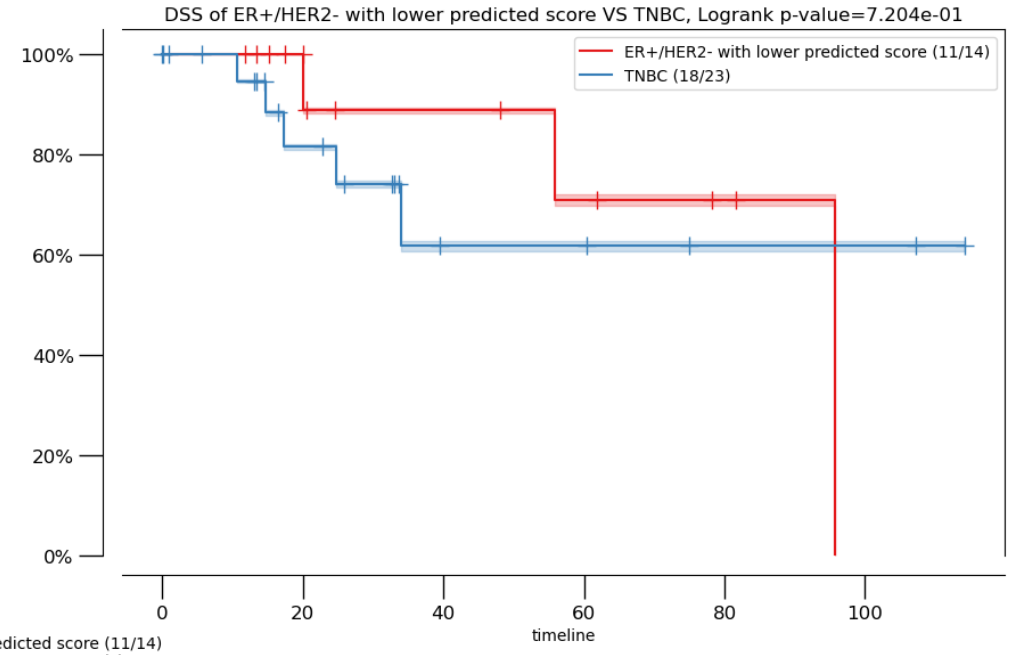
ER+/HER2- with lower predicted score (10/14)						
At risk	14	9	5	4	1	0
Censored	0	3	6	7	9	10
Events	0	2	3	3	4	4
TNBC (18/25)						
At risk	25	13	6	3	2	2
Censored	0	8	14	15	16	16
Events	0	4	5	7	7	7



ER+/HER2- with lower predicted score (10/14)						
At risk	14	10	6	4	2	0
Censored	0	4	7	8	10	10
Events	0	0	1	2	2	4
TNBC (17/25)						
At risk	25	14	6	5	3	2
Censored	0	8	14	14	16	16
Events	0	3	5	6	6	7



ER+/HER2- with lower predicted score (9/11)						
At risk	11	7	4	3	1	0
Censored	0	2	5	6	8	9
Events	0	2	2	2	2	2
TNBC (17/23)						
At risk	23	12	5	2	1	1
Censored	0	8	14	15	16	16
Events	0	3	4	6	6	6



ER+/HER2- with lower predicted score (11/14)						
At risk	14	10	6	4	2	0
Censored	0	4	7	8	10	11
Events	0	0	1	2	2	3
TNBC (18/23)						
At risk	23	12	4	4	2	2
Censored	0	8	14	14	16	16
Events	0	3	5	5	5	5

Evaluation with training data

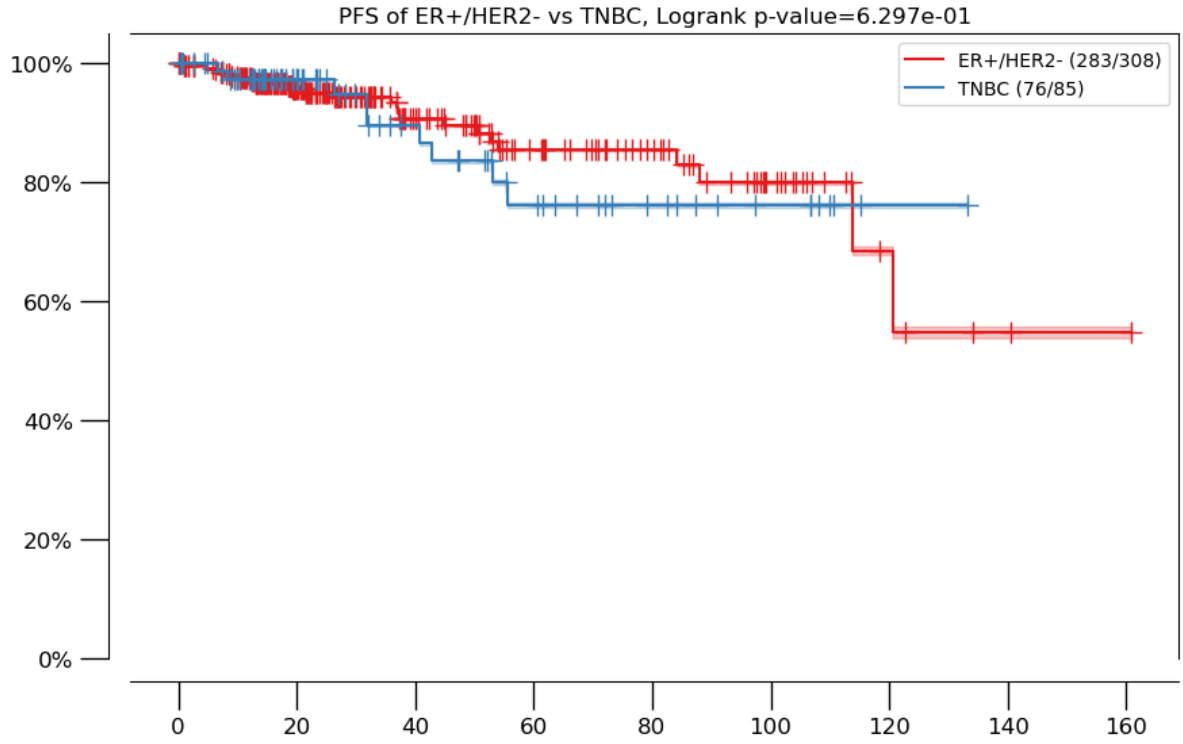
In [37]:
```
cv_train_index = y_train.index
cv_train_predicted_df_dict = {}
import torch.nn.functional as F
for fold in range(5):
    cv_train_predicted_df = pd.DataFrame(np.zeros((len(cv_train_index),2)
    file = np.sort(os.listdir(f"{folder}/{str(fold)}"))[-1] # Get the las
    print(f"Fold {fold} model:", file)
    model_path = f"{folder}/{str(fold)}/{file}"
    checkpoint = torch.load(model_path, map_location=torch.device('cpu'))
    model_ft = models.resnet50().to(device)
    model_test = nn.Sequential(model_ft, net_add).to(device)
    model_test.load_state_dict(checkpoint['model_state_dict'])
    model_test.cuda()
    criterion = nn.CrossEntropyLoss()
    epoch_test = checkpoint['epoch']
    loss_test = checkpoint['loss']
    model_test.eval()

    loss_epoch_test=[]
    y_proba = []
    y_gold = []
    y_pred = []
    with torch.no_grad():
        for b, (X, y) in enumerate(train_dataloader):
            outputs = model_test(X.cuda())
            _, preds = torch.max(outputs, 1)
            y_proba += torch.flatten(outputs[:,1]).cpu().tolist()
            y_gold += y.data.cpu().tolist()
            y_pred += preds.cpu().tolist()
#             loss = criterion(outputs.float(), torch.tensor([[1,0] if la
#             loss_epoch_test.append(loss.item())

        auc=roc_auc_score(y_gold,y_proba)
        print(f"Fold {fold} AUROC: {auc}")
    cv_train_predicted_df.loc[cv_train_index,'score'] = y_proba
    cv_train_predicted_df.loc[cv_train_index,'gold'] = y_gold
    cv_train_predicted_df_dict[fold] = cv_train_predicted_df

# Average the scores of the 5 folds
cv_train_predicted_df["score"] = (cv_train_predicted_df_dict[0]["score"]+
                            cv_train_predicted_df_dict[1]["score"]+\
                            cv_train_predicted_df_dict[2]["score"]+\
                            cv_train_predicted_df_dict[3]["score"]+\
                            cv_train_predicted_df_dict[4]["score"])/5
```

```
Fold 0 model: epoch_21_loss_0.617512_auroc_0.729207.pt
Fold 0 AUROC: 0.6942643229166666
Fold 1 model: epoch_9_loss_0.674120_auroc_0.596405.pt
Fold 1 AUROC: 0.6580078125
Fold 2 model: epoch_6_loss_0.676420_auroc_0.628654.pt
Fold 2 AUROC: 0.6107552083333334
Fold 3 model: epoch_9_loss_0.650217_auroc_0.667819.pt
Fold 3 AUROC: 0.671875
Fold 4 model: epoch_8_loss_0.667883_auroc_0.647039.pt
Fold 4 AUROC: 0.6402083333333334
```

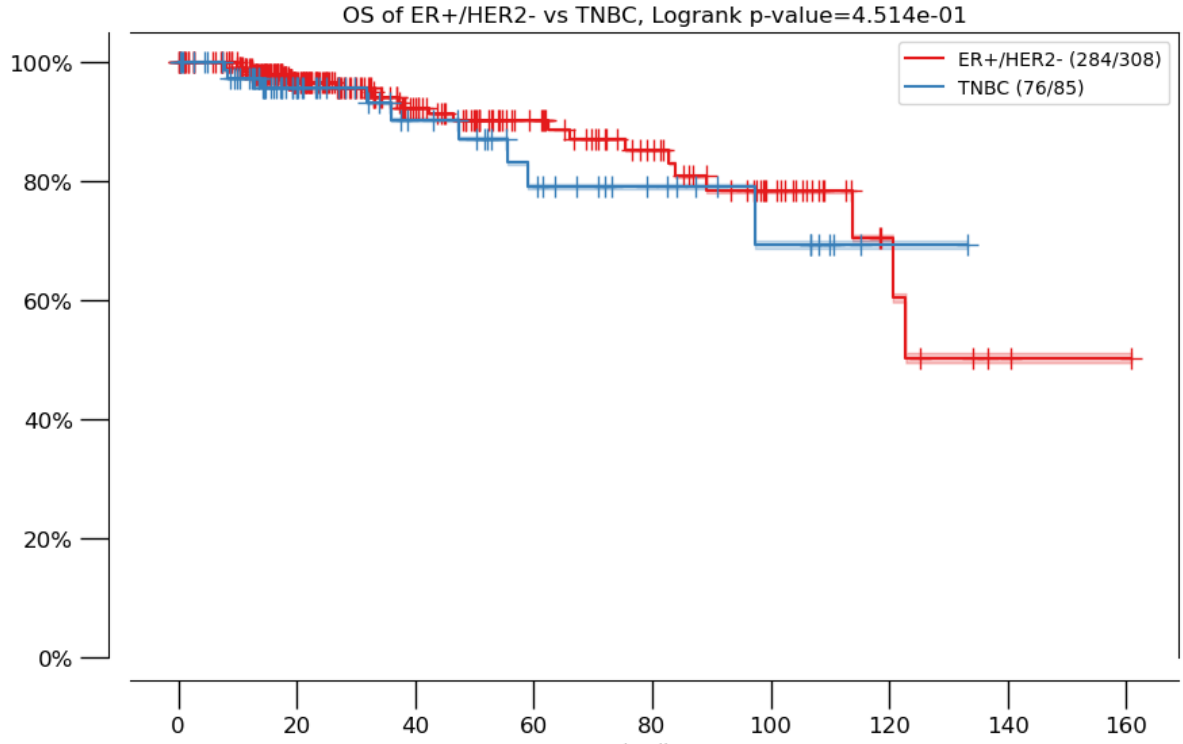
Evluation of the survival of the training data

In [30]:
```
cv_train_predicted_df_ihc = cv_train_predicted_df.join(meta_erihc, how='i
cv_train_predicted_df_ihc["Subtype"] = None
cv_train_predicted_df_ihc["Subtype"][(cv_train_predicted_df_ihc["er_statu
# cv_predicted_df_ihc["Subtype"][cv_predicted_df_ihc["her2_status_by_ihc"
cv_train_predicted_df_ihc["Subtype"][(cv_train_predicted_df_ihc["er_statu
cv_train_predicted_df_ihc["EERES"] = brca_earlyes_es.loc[cv_train_predict
cv_train_predicted_df_ihc["ESR1"] = brca_lv3.loc[cv_train_predicted_df_ih

brca_clinical_training = brca_clinical.join(cv_train_predicted_df_ihc, ho
brca_clinical_training_dfs = brca_clinical_training[["DFS_MONTHS", "DFS_S
brca_clinical_training_dss = brca_clinical_training[["DSS_MONTHS", "DSS_S
brca_clinical_training_pfs = brca_clinical_training[["PFS_MONTHS", "PFS_S
brca_clinical_training_os = brca_clinical_training[["OS_MONTHS", "OS_STAT

result = km.fit(brca_clinical_training_pfs['PFS_MONTHS'], brca_clinical_t
km.plot(result, title=f"PFS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_training_os['OS_MONTHS'], brca_clinical_tra
km.plot(result, title=f"OS of ER+/HER2- vs TNBC, Logrank p-value={result[
plt.show()
result = km.fit(brca_clinical_training_dfs['DFS_MONTHS'], brca_clinical_t
km.plot(result, title=f"DFS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()
result = km.fit(brca_clinical_training_dss['DSS_MONTHS'], brca_clinical_t
km.plot(result, title=f"DSS of ER+/HER2- vs TNBC, Logrank p-value={result
plt.show()


plt.scatter(cv_train_predicted_df_ihc[cv_train_predicted_df_ihc["Subtype"
# plt.scatter(cv_predicted_df_ihc[cv_predicted_df_ihc["Subtype"]=="ERBB2+
plt.scatter(cv_train_predicted_df_ihc[cv_train_predicted_df_ihc["Subtype"
plt.legend(["ER+/HER2-", "TNBC"])
plt.xlabel("ESR1")
plt.ylabel("EERES")
plt.show()
```

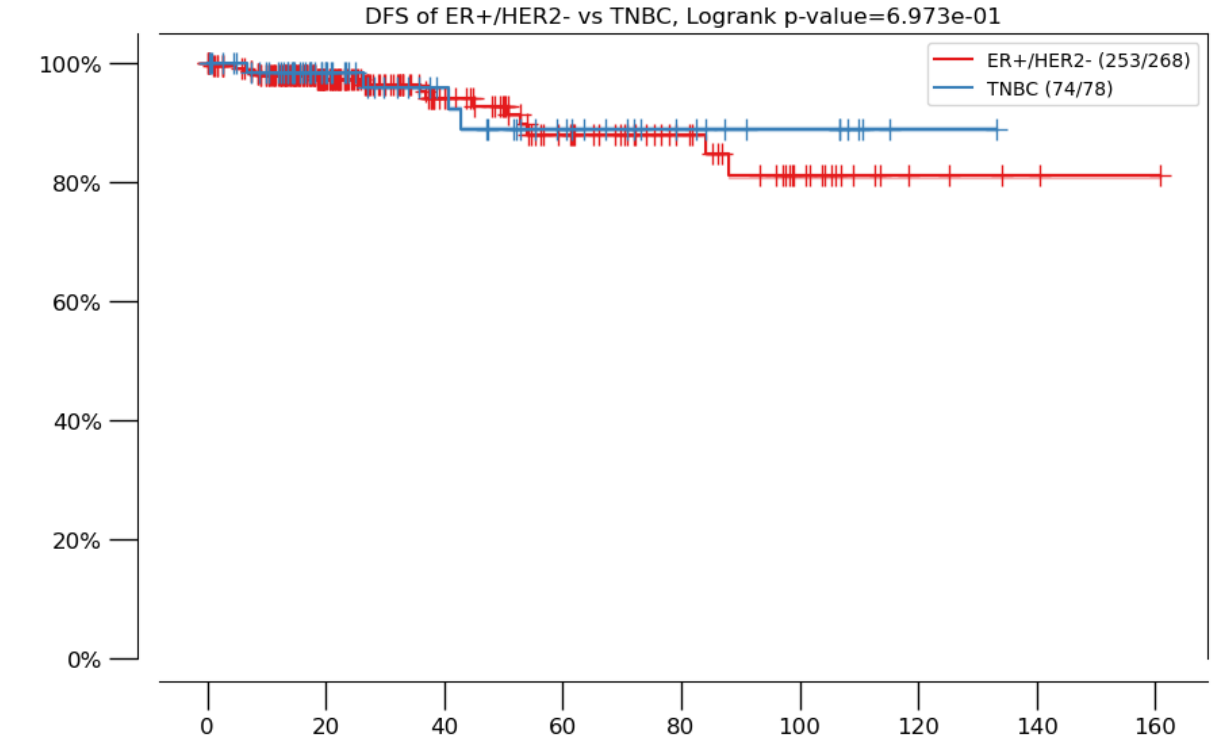
```
/tmp/ipykernel_93586/1740958595.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cv_train_predicted_df_ihc["Subtype"][(cv_train_predicted_df_ihc["er_st
atus_by_ihc"]=='Positive') & (cv_train_predicted_df_ihc["her2_status_by_
ihc"]=='Negative')] = "ER+/HER2-"
/tmp/ipykernel_93586/1740958595.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cv_train_predicted_df_ihc["Subtype"][(cv_train_predicted_df_ihc["er_st
atus_by_ihc"]=='Negative') & (cv_train_predicted_df_ihc["her2_status_by_
ihc"]=='Negative') & (cv_train_predicted_df_ihc["pr_status_by_ihc"]=='Ne
gative')] = "TNBC"
```



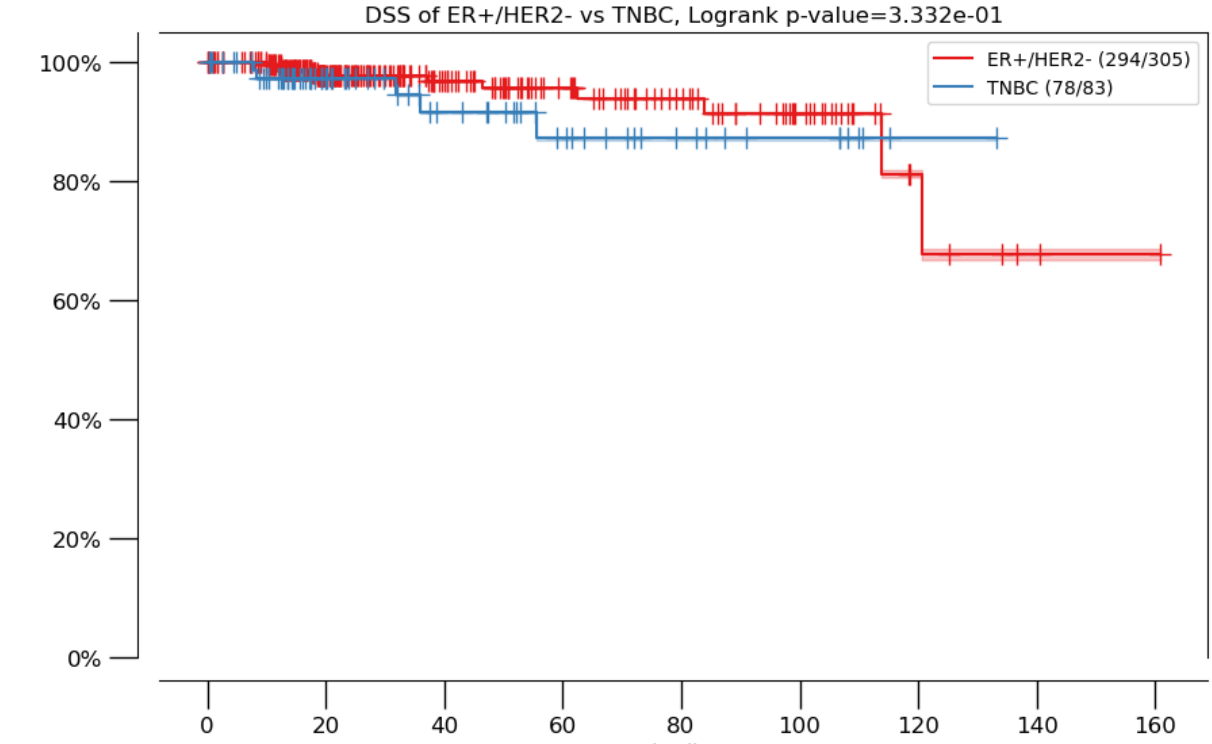
ER+/HER2- (283/308)									
At risk	302	166	90	56	37	18	5	2	1
Censored	6	131	201	231	250	267	279	281	282
Events	0	11	17	21	21	23	24	25	25
TNBC (76/85)									
At risk	85	47	30	20	12	7	1	0	0
Censored	0	36	50	56	64	69	75	76	76
Events	0	2	5	9	9	9	9	9	9



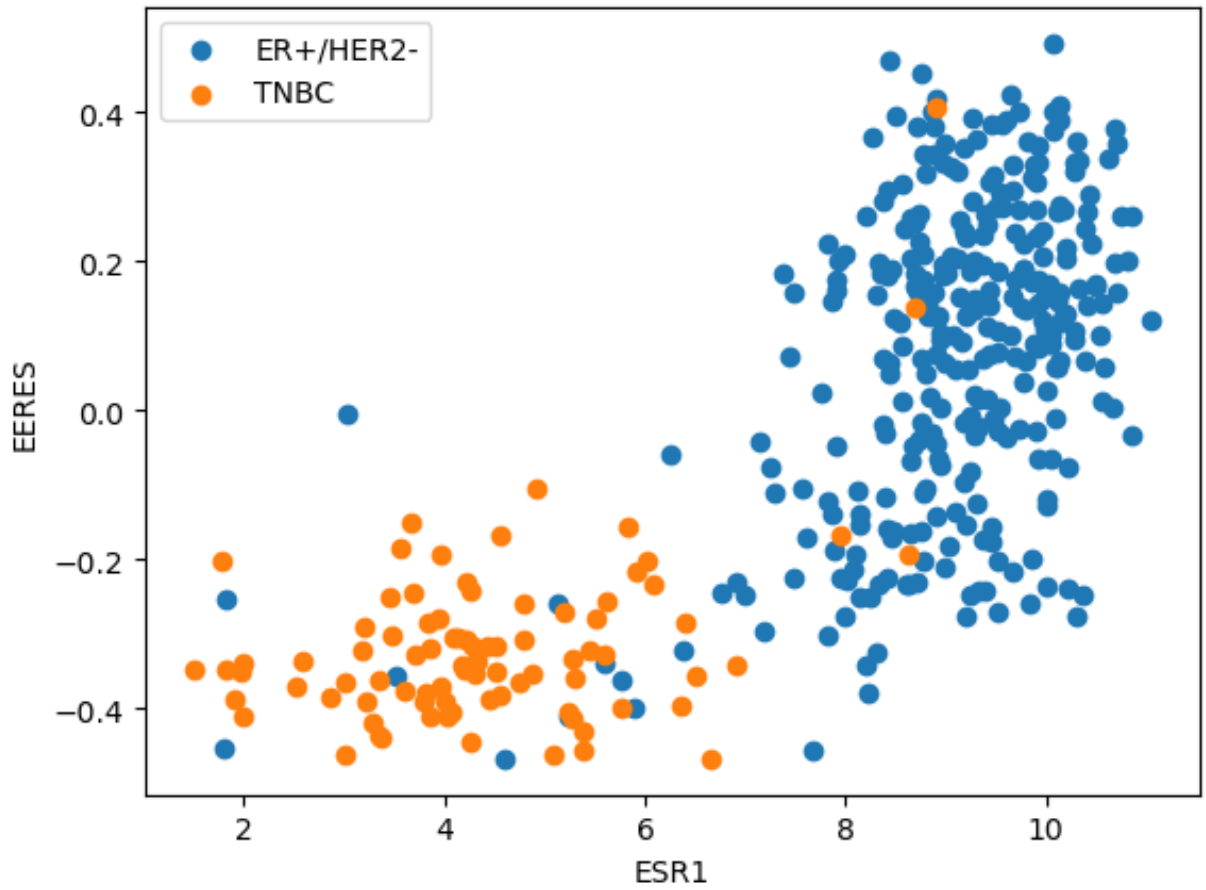
ER+/HER2- (284/308)									
At risk	302	171	95	63	42	23	7	2	1
Censored	6	129	200	230	248	264	279	282	283
Events	0	8	13	15	18	21	22	24	24
TNBC (76/85)									
At risk	85	47	30	20	12	7	1	0	0
Censored	0	35	50	57	65	69	75	76	76
Events	0	3	5	8	8	9	9	9	9



ER+/HER2- (253/268)									
At risk	263	143	77	48	29	15	4	2	1
Censored	5	119	182	207	226	238	249	251	252
Events	0	6	9	13	13	15	15	15	15
TNBC (74/78)									
At risk	78	44	28	19	11	7	1	0	0
Censored	0	33	48	55	63	67	73	74	74
Events	0	1	2	4	4	4	4	4	4



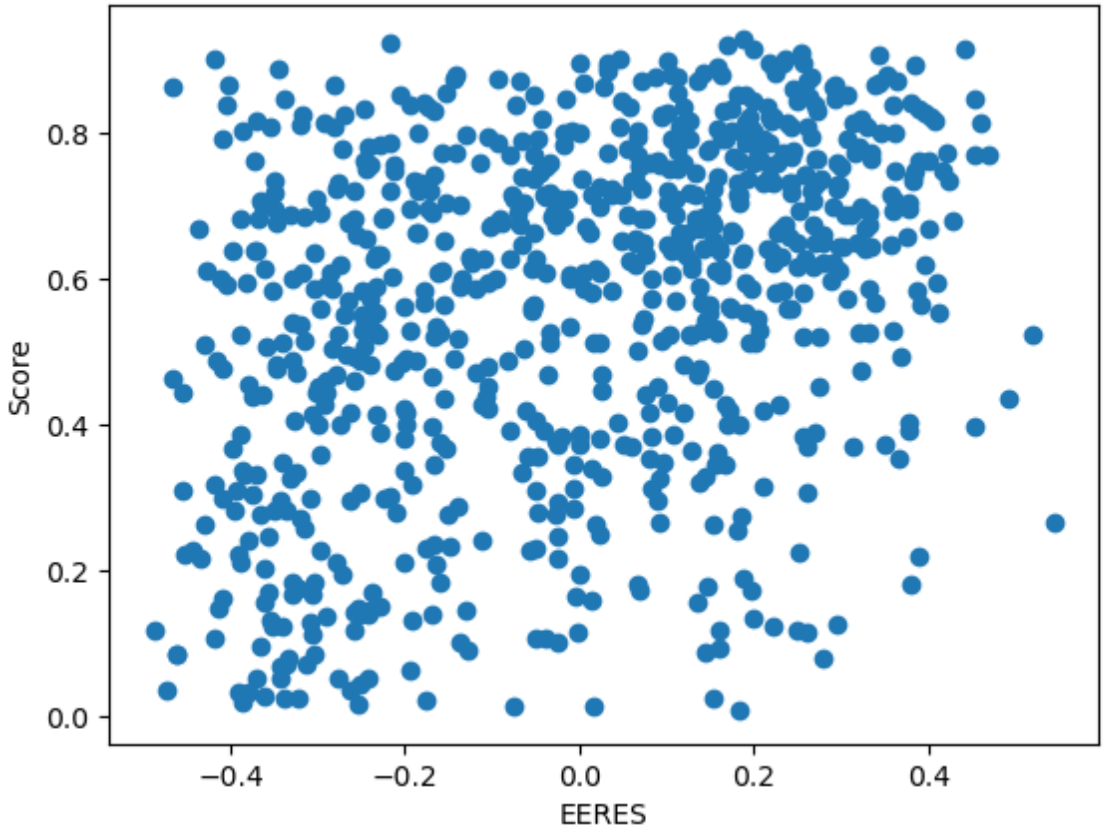
ER+/HER2- (294/305)									
At risk	299	168	94	62	41	22	6	2	1
Censored	6	132	205	236	256	274	289	292	293
Events	0	5	6	7	8	9	10	11	11
TNBC (78/83)									
At risk	83	46	29	19	11	7	1	0	0
Censored	0	35	50	59	67	71	77	78	78
Events	0	2	4	5	5	5	5	5	5

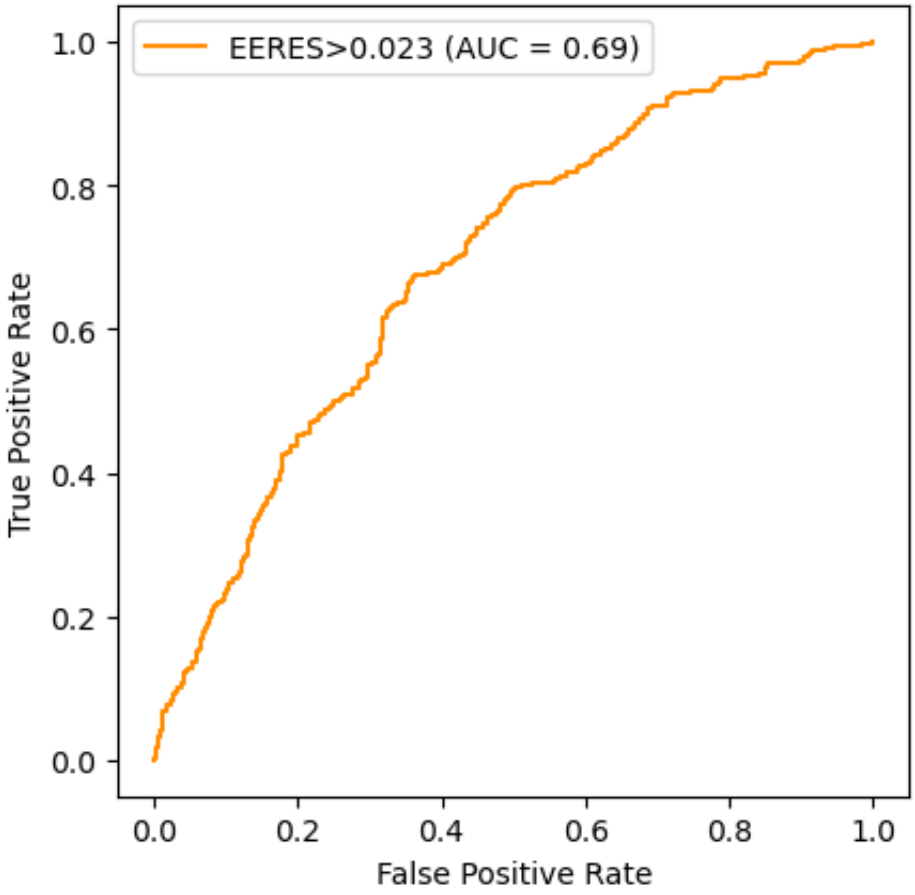


In [32]:

```
plt.scatter(cv_train_predicted_df_ihc["EERES"], cv_train_predicted_df_ihc
plt.xlabel("EERES")
plt.ylabel("Score")
r, p = pearsonr(cv_train_predicted_df_ihc["EERES"], cv_train_predicted_df
plt.title(f"Pearson R: {r}, p-value: {p}")

auroc = roc_auc_score(cv_train_predicted_df_ihc["gold"], cv_train_predict
RocCurveDisplay.from_predictions(
    cv_train_predicted_df_ihc["gold"],
    cv_train_predicted_df_ihc["score"],
    name=f"EERES>{eeres_threshold:.3f}",
    color="darkorange",
)
# plt.plot([0, 1], [0, 1], "k--", label="chance level (AUC = 0.5)")
plt.axis("square")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
# plt.title("Receiver Operating Characteristic")
plt.legend()
plt.show()
```

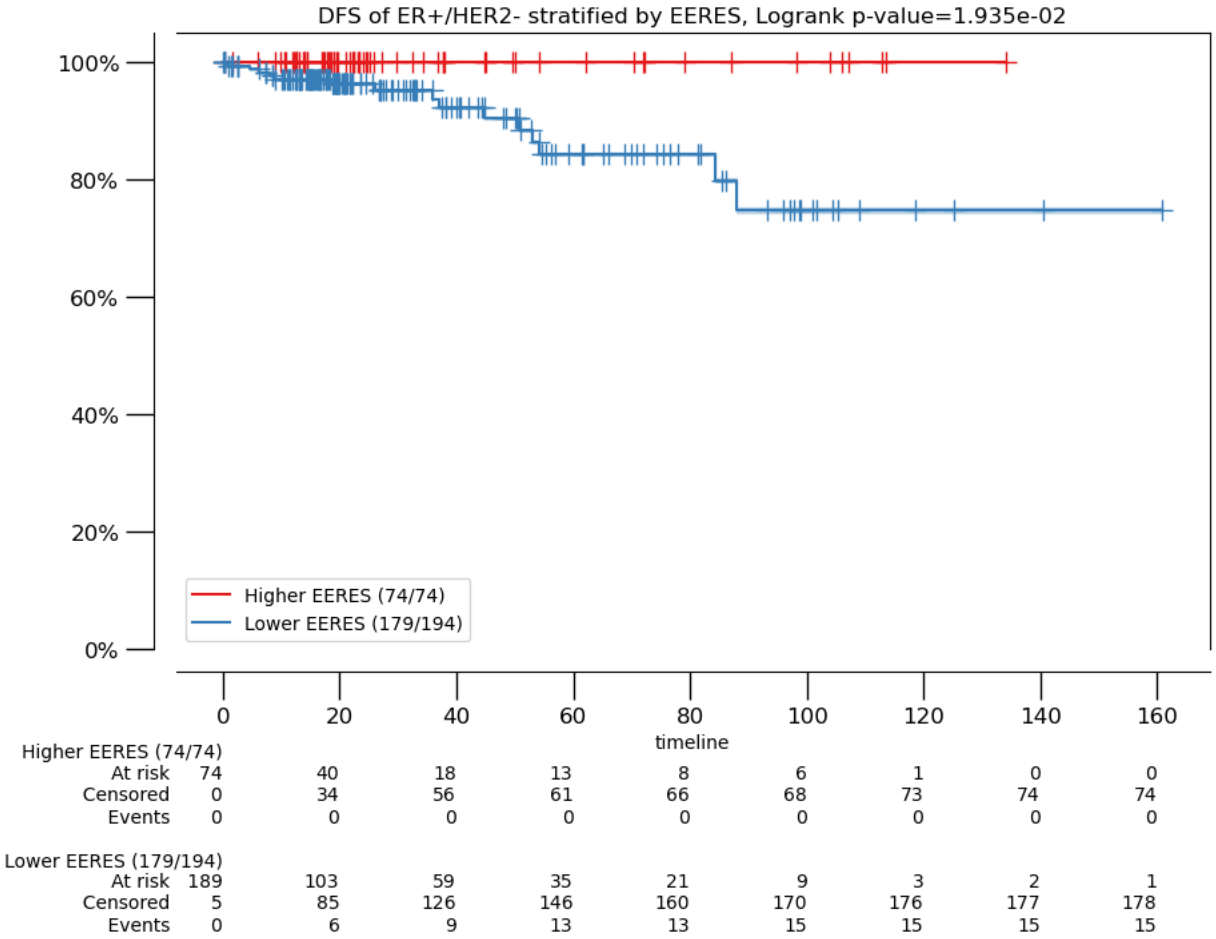
Pearson R: 0.36641120844482955, p-value: 2.5514554355708225e-26



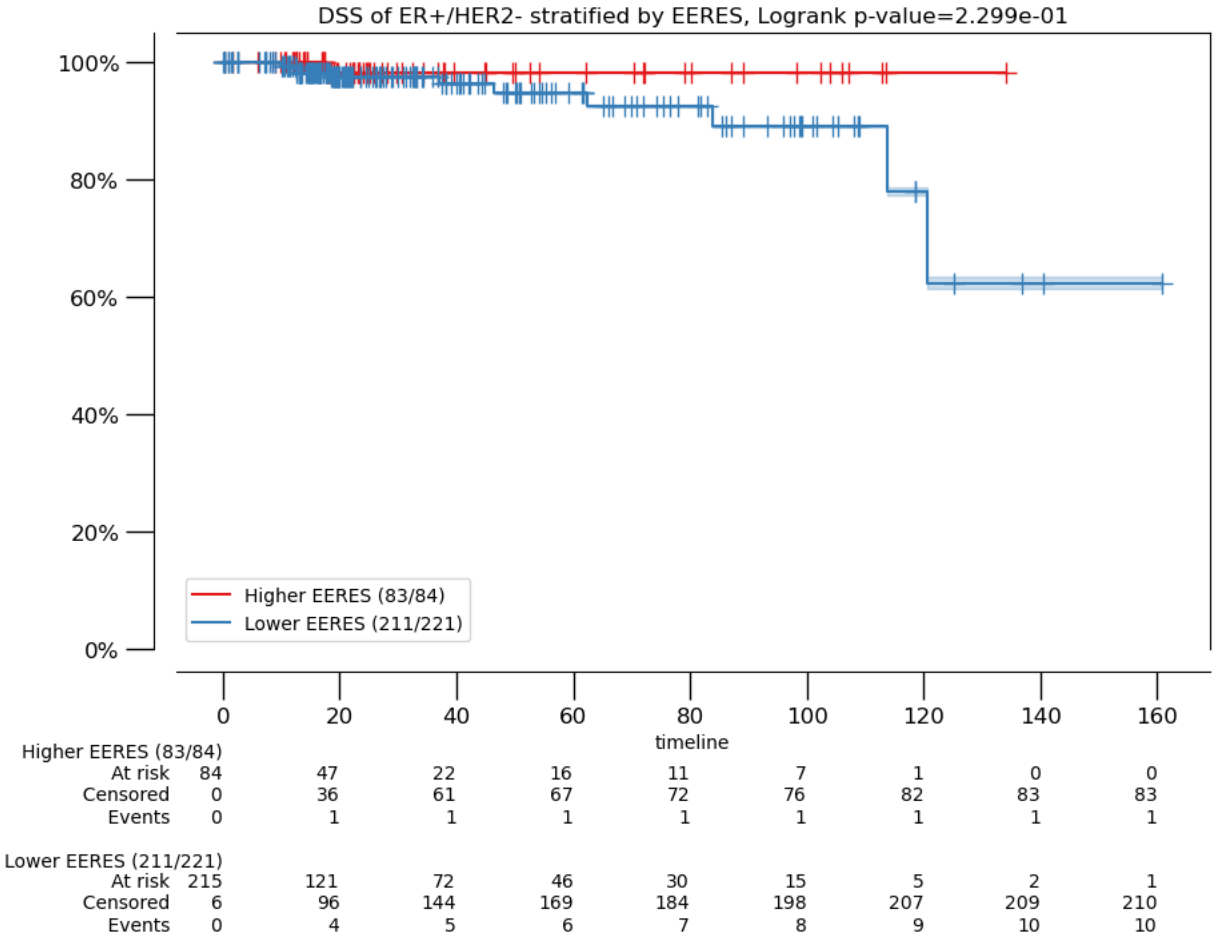


In [36]:
```
brca_clinical_training_erposerbb2neg = brca_clinical.join(cv_train_predic
brca_clinical_training_erposerbb2neg

brca_clinical_training_erposerbb2neg_dfs = brca_clinical_training_erposer
brca_clinical_training_erposerbb2neg_dss = brca_clinical_training_erposer
brca_clinical_training_erposerbb2neg_pfs = brca_clinical_training_erposer
brca_clinical_training_erposerbb2neg_os = brca_clinical_training_erposerb
finding_best_threshold_with_FS_SS(brca_clinical_training_erposerbb2neg_df
finding_best_threshold_with_FS_SS(brca_clinical_training_erposerbb2neg_pf
finding_best_threshold_with_FS_SS(brca_clinical_training_erposerbb2neg_df
finding_best_threshold_with_FS_SS(brca_clinical_training_erposerbb2neg_pf
```

```
Best EERES threshold: 0.20434858171014614
```

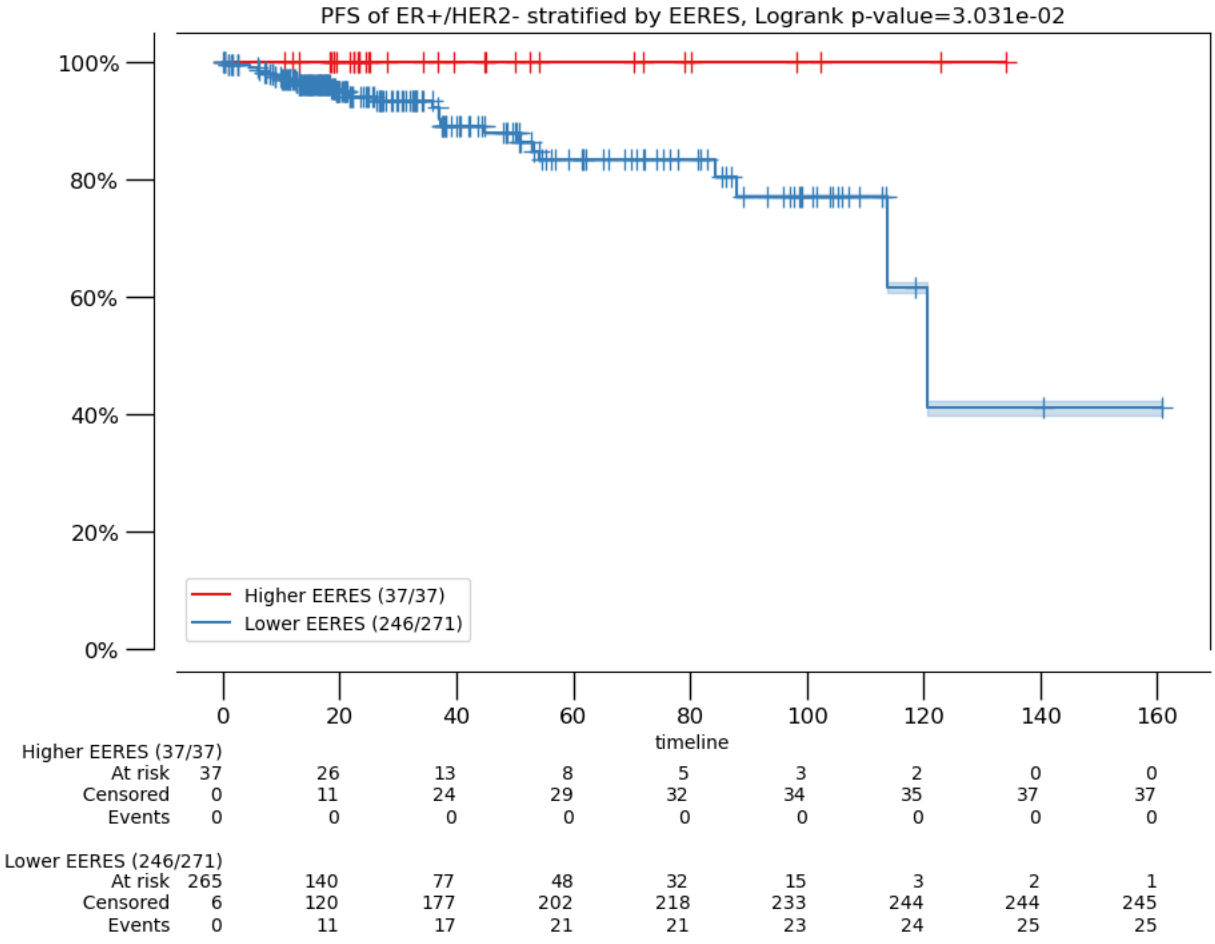


Higher EERES (74/74)									
At risk	74	40	18	13	8	6	1	0	0
Censored	0	34	56	61	66	68	73	74	74
Events	0	0	0	0	0	0	0	0	0
Lower EERES (179/194)									
At risk	189	103	59	35	21	9	3	2	1
Censored	5	85	126	146	160	170	176	177	178
Events	0	6	9	13	13	15	15	15	15

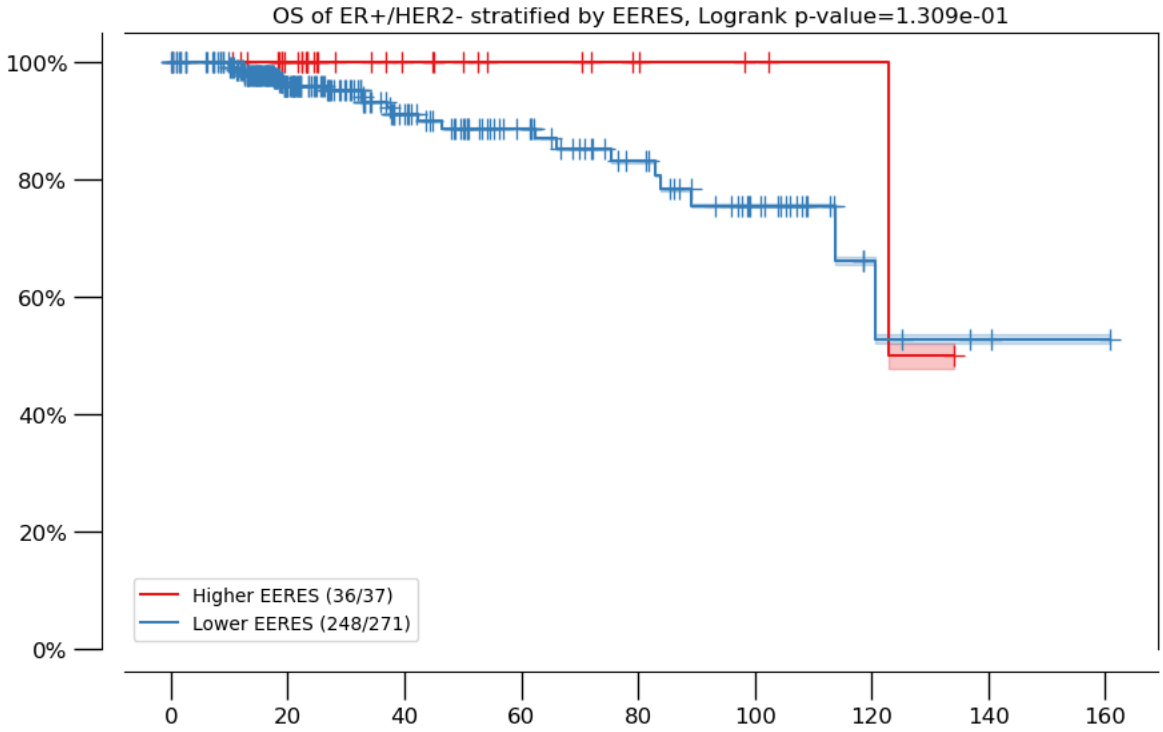


Higher EERES (83/84)									
At risk	84	47	22	16	11	7	1	0	0
Censored	0	36	61	67	72	76	82	83	83
Events	0	1	1	1	1	1	1	1	1
Lower EERES (211/221)									
At risk	215	121	72	46	30	15	5	2	1
Censored	6	96	144	169	184	198	207	209	210
Events	0	4	5	6	7	8	9	10	10

Best EERES threshold: 0.32551106034849975

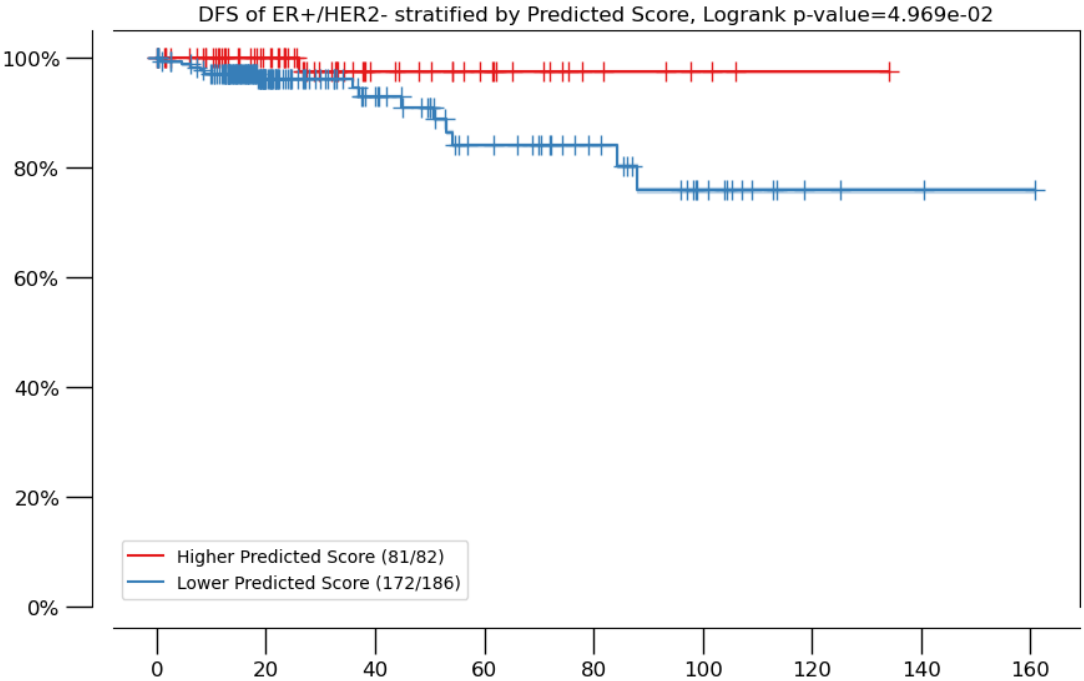


Higher EERES (37/37)									
At risk	37	26	13	8	5	3	2	0	0
Censored	0	11	24	29	32	34	35	37	37
Events	0	0	0	0	0	0	0	0	0
Lower EERES (246/271)									
At risk	265	140	77	48	32	15	3	2	1
Censored	6	120	177	202	218	233	244	244	245
Events	0	11	17	21	21	23	24	25	25

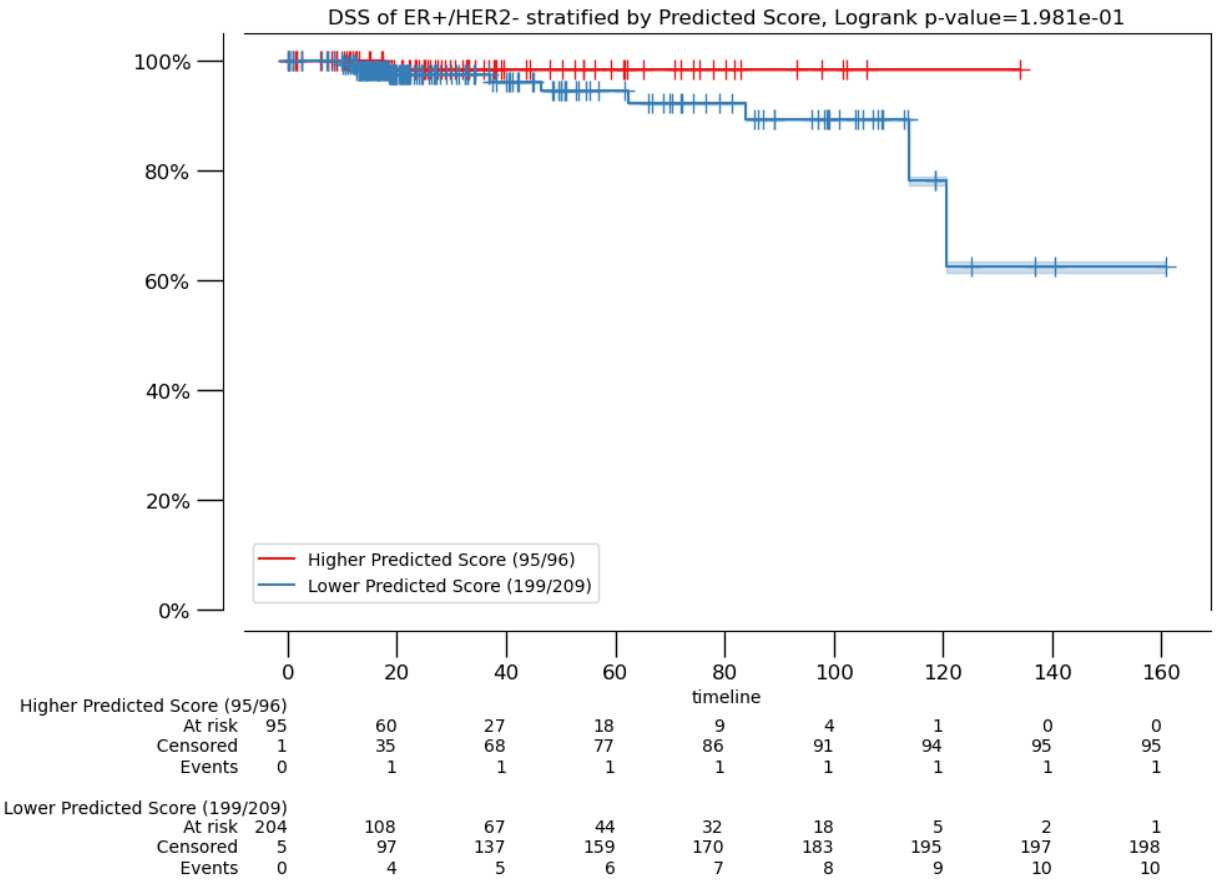


Higher EERES (36/37)									
At risk	37	26	13	8	5	3	2	0	0
Censored	0	11	24	29	32	34	35	36	36
Events	0	0	0	0	0	0	0	1	1
Lower EERES (248/271)									
At risk	265	145	82	55	37	20	5	2	1
Censored	6	118	176	201	216	230	244	246	247
Events	0	8	13	15	18	21	22	23	23

Best score threshold: 0.7400792701914911



Higher Predicted Score (81/82)									
At risk	81	50	23	15	6	3	1	0	0
Censored	1	32	58	66	75	78	80	81	81
Events	0	0	1	1	1	1	1	1	1
Lower Predicted Score (172/186)									
At risk	182	93	54	33	23	12	3	2	1
Censored	4	87	124	141	151	160	169	170	171
Events	0	6	8	12	12	14	14	14	14



Higher Predicted Score (95/96)									
At risk	95	60	27	18	9	4	1	0	0
Censored	1	35	68	77	86	91	94	95	95
Events	0	1	1	1	1	1	1	1	1
Lower Predicted Score (199/209)									
At risk	204	108	67	44	32	18	5	2	1
Censored	5	97	137	159	170	183	195	197	198
Events	0	4	5	6	7	8	9	10	10

not significant