Photon and Electron classification for endcap region (EE) with and without tracking information

In [1]:
```
import uproot
import numpy as np
import pandas as pd
import h5py
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.layers import Input, Activation, Dense, Convolution2D, MaxPooling2D, Dropout, Flatten
from tensorflow import keras
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

Reading the root file with uprood and coverting them to pandas data frame

In [2]:

```
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)

treename = 'fTree' # this is the name of the tree in the root file
filename = {}
upfile = {}
df = {}


filename['photon'] = 'data/Signal.root' # this the the file which contains photons (diphotons)
filename['electron'] = 'data/Background.root' # this is the file which contains electrons and positron.

branches1 = ['Photon1']


upfile['photon1'] = uproot.open(filename['photon'])
upfile['electron1'] = uproot.open(filename['electron'])
df['photon1'] = upfile['photon1'][treename].arrays(branches1, library='pd')
df['electron1'] = upfile['electron1'][treename].arrays(branches1, library='pd')
df['photon1']['isPhoton'] = np.ones(len(df['photon1']))
df['electron1']['isPhoton'] = np.zeros(len(df['electron1']))

branches2 = ['Photon2']


upfile['photon2'] = uproot.open(filename['photon'])
upfile['electron2'] = uproot.open(filename['electron'])
df['photon2'] = upfile['photon2'][treename].arrays(branches2, library='pd')
df['electron2'] = upfile['electron2'][treename].arrays(branches2, library='pd')
df['photon2']['isPhoton'] = np.ones(len(df['photon2']))
df['electron2']['isPhoton'] = np.zeros(len(df['electron2']))
```

Selecting Photon and electron which passes certain condition and creating separate data frame for them, this data frame is not used for training purpose. This is only for analysis of data

In [3]:
```
df['electron1_pashipt'] = df['electron1'][(df['electron1'][('Photon1','pt')] >= 30.0)
    & (df['electron1'][('Photon1','pt')] <= 1800)
    & (df['electron1'][('Photon1','isEE')] ==1) & (df['electron1'][('Photon1','passHighPtID')] == 1)]
# len(df['electron1_pashipt'])

df['photon1_pashipt'] = df['photon1'][(df['photon1'][('Photon1','pt')] >= 30.0) &
                              (df['photon1'][('Photon1','pt')] <= 1800)
     & (df['photon1'][('Photon1','isEE')] ==1) & (df['photon1'][('Photon1','passHighPtID')] ==1)]
# len(df['photon1_pashipt'])

df['electron2_pashipt'] = df['electron2'][(df['electron2'][('Photon2','pt')] >= 30.0) & (df['electron2'][('Photon2','pt')] <= 1800)
     & (df['electron2'][('Photon2','isEE')] ==1) & (df['electron2'][('Photon2','passHighPtID')] ==1)]
# len(df['electron2_pashipt'])

df['electron2_pashipt'] = df['electron2'][(df['electron2'][('Photon2','pt')] >= 30.0) &
                                  (df['electron2'][('Photon2','pt')] <= 1800)
     & (df['electron2'][('Photon2','isEE')] ==1) & (df['electron2'][('Photon2','passHighPtID')] ==1)]
# len(df['electron2_pashipt'])
```

Selecting the particles with high transeve momentum in EE region

In [4]:
```
df['electron1'] = df['electron1'][(df['electron1'][('Photon1','pt')] >= 30.0) &
                (df['electron1'][('Photon1','pt')] <= 1800) & (df['electron1'][('Photon1','isEE')] ==1)]

df['photon1'] = df['photon1'][(df['photon1'][('Photon1','pt')] >= 30.0) &
                (df['photon1'][('Photon1','pt')] <= 1800) & (df['photon1'][('Photon1','isEE')] ==1)]

df['electron2'] = df['electron2'][(df['electron2'][('Photon2','pt')] >= 30.0) &
                (df['electron2'][('Photon2','pt')] <= 1800) & (df['electron2'][('Photon2','isEE')] ==1)]


df['photon2'] = df['photon2'][(df['photon2'][('Photon2','pt')] >= 30.0) &
                (df['photon2'][('Photon2','pt')] <= 1800) & (df['photon2'][('Photon2','isEE')] ==1)]
```

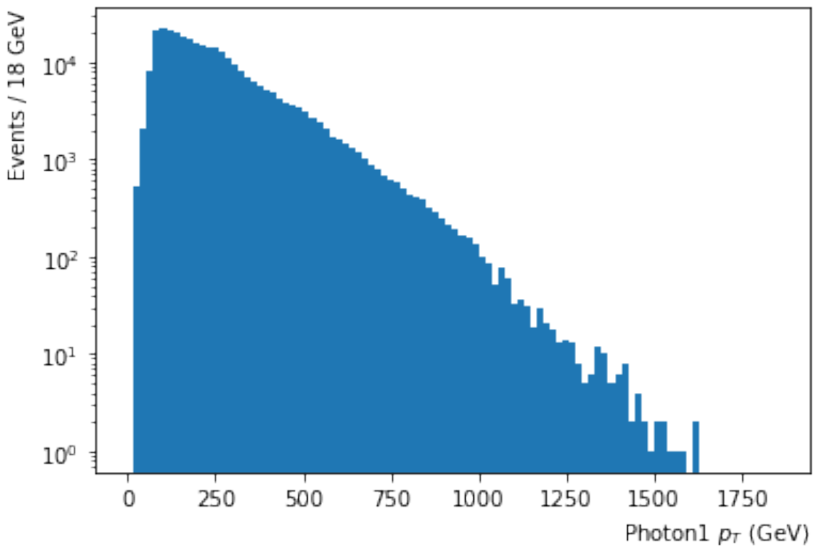
In [5]:
```
# df["photon2"]['isPhoton']
```

In [6]:
```
# df['electron1'].isnull().sum().sum()
```

Lets plot few parameters

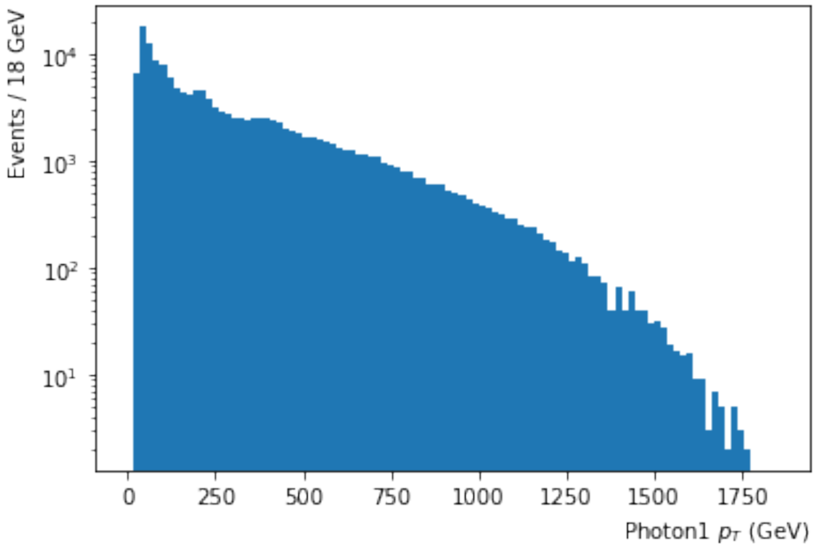
In [7]:

```
n_bins_pt = 100;
df['photon1'][('Photon1','pt')].plot.hist(bins=n_bins_pt,range=(0,1850),log=1);
plt.xlabel('Photon1 $p_T$ (GeV)', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(int(1850/n_bins_pt))+' GeV', horizontalalignment='right', y=1.0);
```



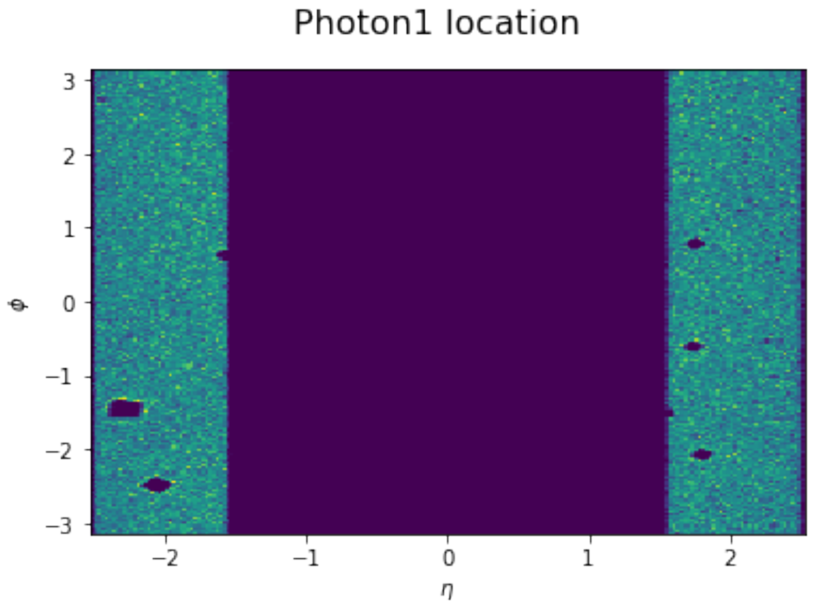
In [8]:

```
n_bins_pt = 100;
df['electron1'][('Photon1','pt')].plot.hist(bins=n_bins_pt,range=(0,1850),log=1);
plt.xlabel('Photon1 $p_T$ (GeV)', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(int(1850/n_bins_pt))+' GeV', horizontalalignment='right', y=1.0);
```



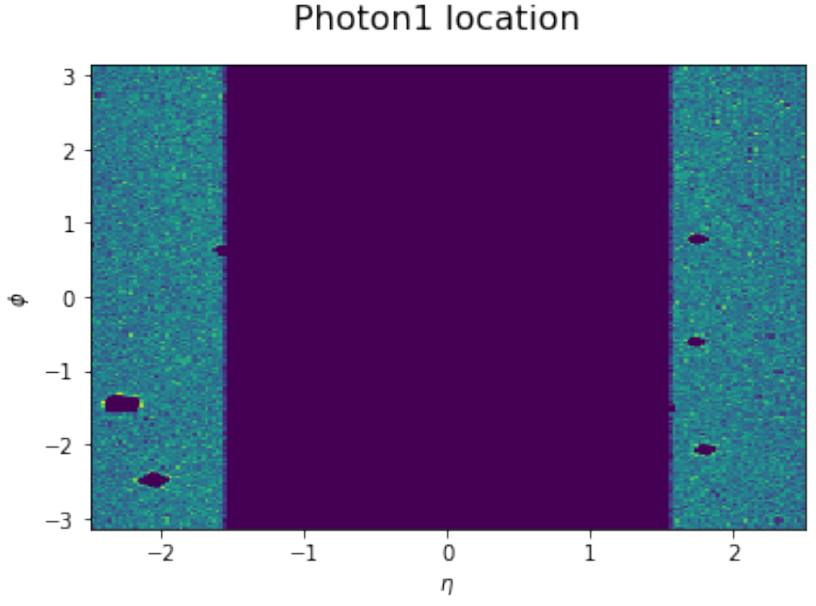
In [9]:

```
plt.hist2d(df['photon1'][('Photon1','eta')],df['photon1'][('Photon1','phi')],bins=200);
plt.suptitle('Photon1 location',fontsize=16)
plt.xlabel('$\eta$');
plt.ylabel('$\phi$');
```

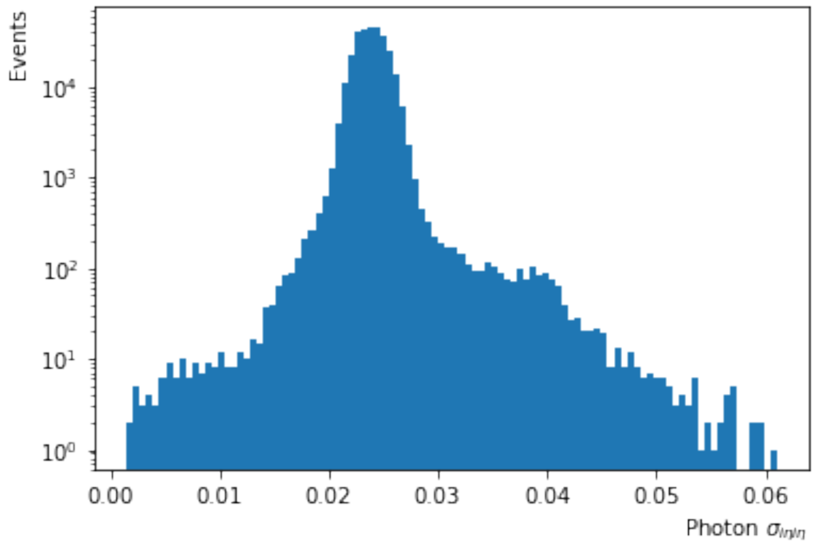


In [10]:

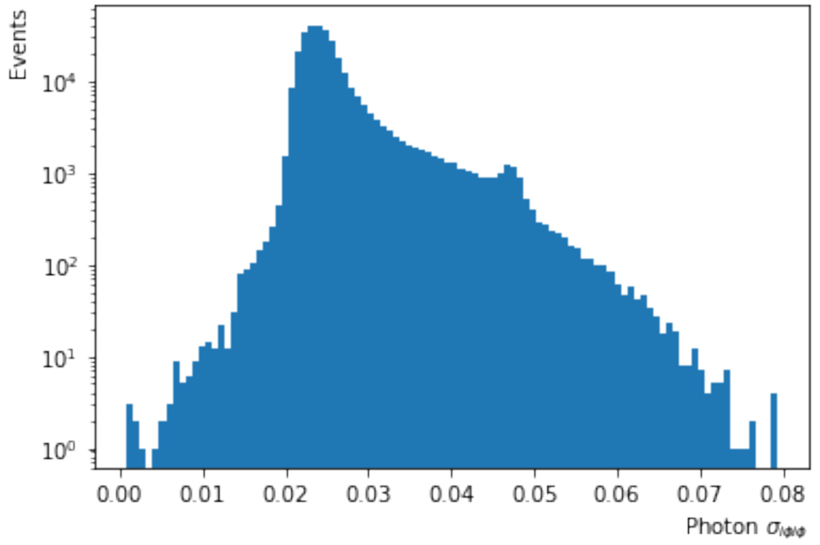
```
plt.hist2d(df['photon1'][('Photon1','scEta')],df['photon1'][('Photon1','scPhi')],bins=200);
plt.suptitle('Photon1 location',fontsize=16)
plt.xlabel('$\eta$');
plt.ylabel('$\phi$');
```



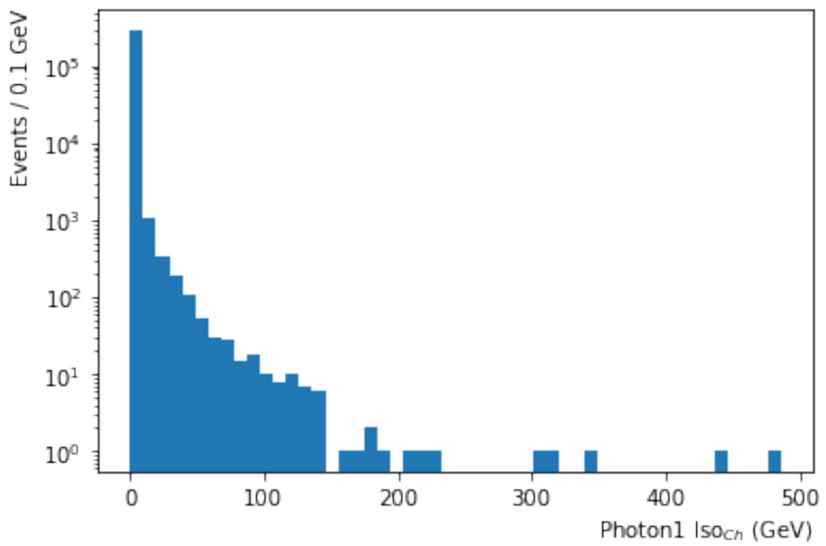
In [11]:
```
df['photon1'][('Photon1','sigmaIetaIeta5x5')].plot.hist(bins=100,log=1);
plt.xlabel('Photon $\sigma_{i\eta i\eta}$', horizontalalignment='right', x=1.0);
plt.ylabel('Events', horizontalalignment='right', y=1.0);
```



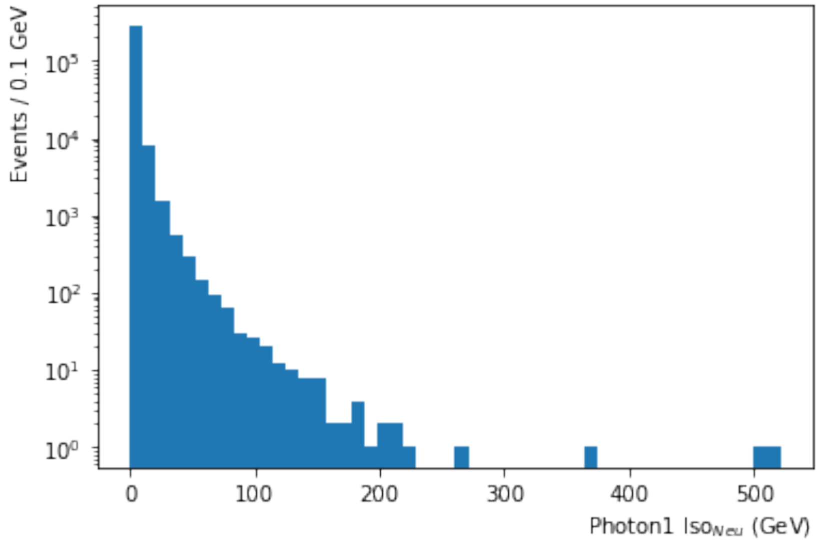
In [12]:
```
df['photon1'][('Photon1','sigmaIphiIphi5x5')].plot.hist(bins=100,log=1);
plt.xlabel('Photon $\sigma_{i\phi i\phi}$', horizontalalignment='right', x=1.0);
plt.ylabel('Events', horizontalalignment='right', y=1.0);
```



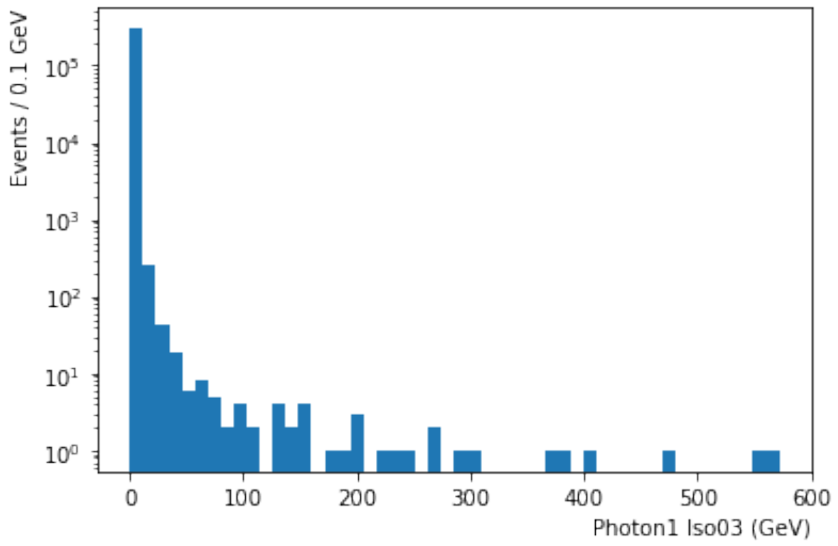
In [13]:
```
df['photon1'][('Photon1','chargedHadIso03')].plot.hist(bins=50,log=True);
plt.xlabel('Photon1 Iso$_{Ch}$ (GeV)', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(5/50)+' GeV', horizontalalignment='right', y=1.0);
```



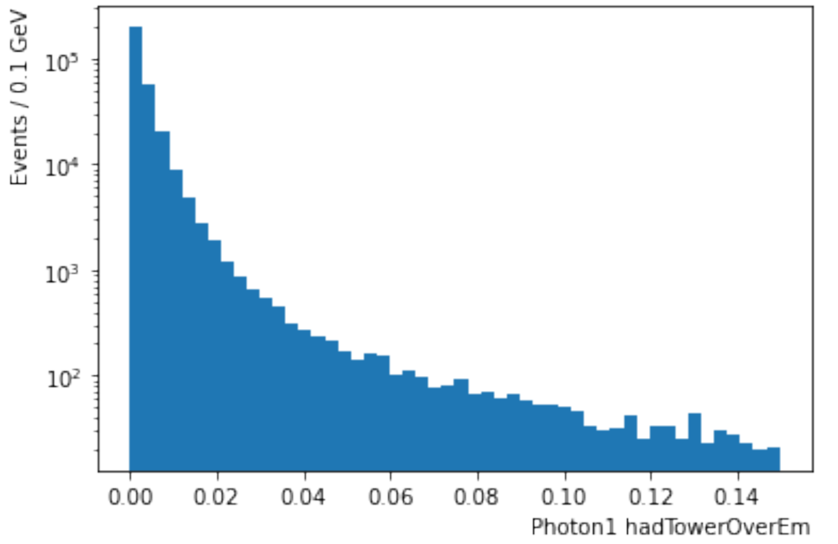
In [14]:
```
df['photon1'][('Photon1','neutralHadIso03')].plot.hist(bins=50,log=True);
plt.xlabel('Photon1 Iso$_{Neu}$ (GeV)', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(5/50)+' GeV', horizontalalignment='right', y=1.0);
```



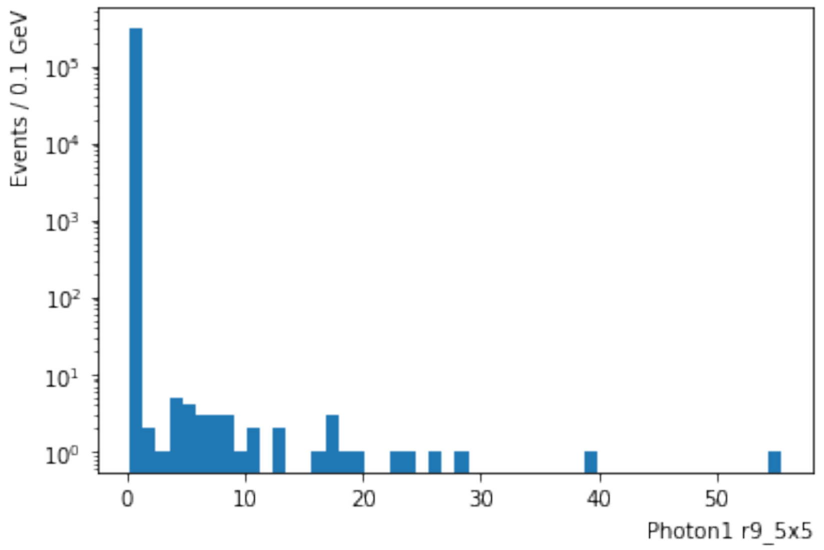
In [15]:
```
df['photon1'][('Photon1','photonIso03')].plot.hist(bins=50,log=True);
plt.xlabel('Photon1 Iso03 (GeV)', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(5/50)+' GeV', horizontalalignment='right', y=1.0);
```



In [16]:
```
df['photon1'][('Photon1','hadTowerOverEm')].plot.hist(bins=50,log=True);
plt.xlabel('Photon1 hadTowerOverEm', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(5/50)+' GeV', horizontalalignment='right', y=1.0);
```

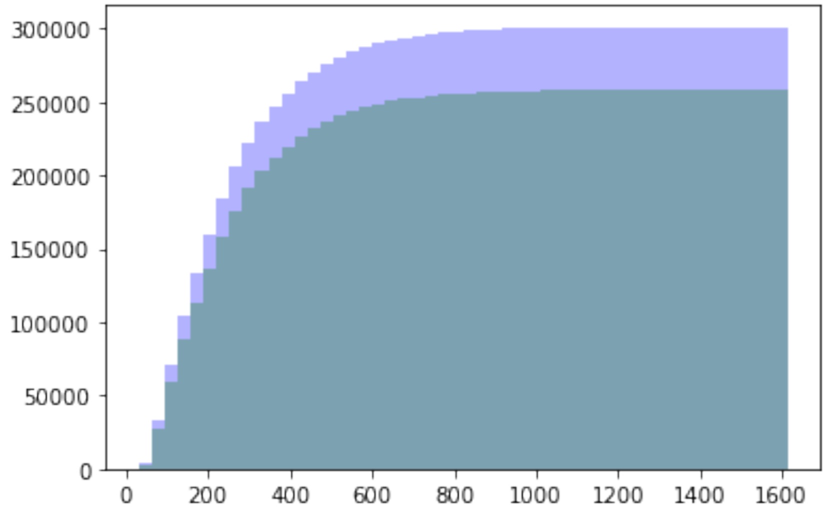


In [17]:
```
df['photon1'][('Photon1','r9_5x5')].plot.hist(bins=50,log=True);
plt.xlabel('Photon1 r9_5x5', horizontalalignment='right', x=1.0);
plt.ylabel('Events / '+str(5/50)+' GeV', horizontalalignment='right', y=1.0);
```

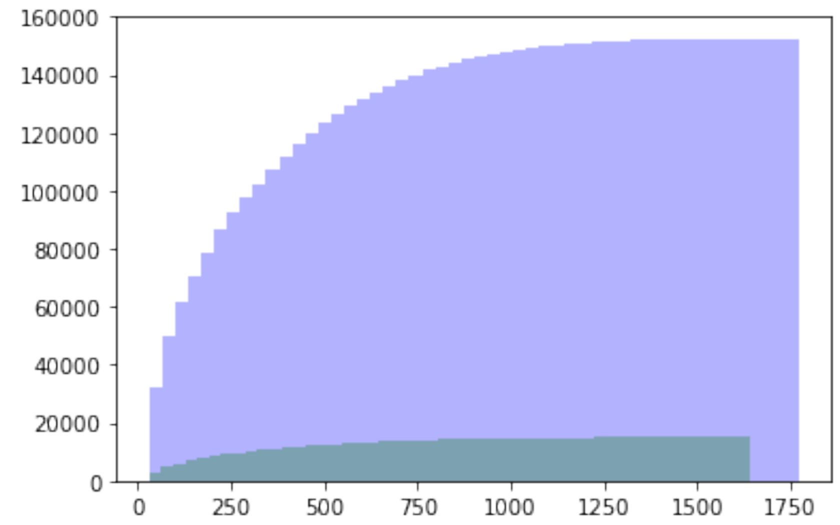


Few plot for efficiency calculation does not related to this project

In [18]:
```
n1,bins1,patches1 = plt.hist(df['photon1'][('Photon1','pt')],bins=50,alpha=0.3,color='blue',cumulative=1,density=0)
n2,bins1,patches1 = plt.hist(df['photon1_pashipt'][('Photon1','pt')],bins=50,alpha=0.3,color='green',cumulative=1,density=0)
```

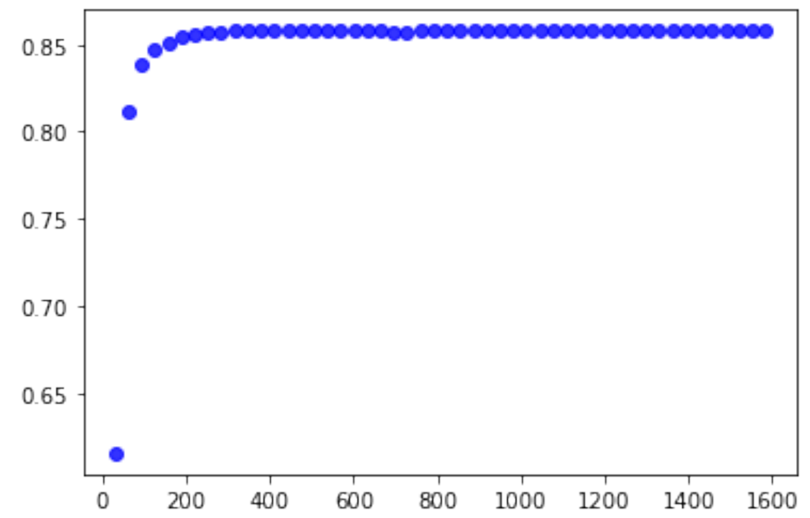


In [19]:
```
n3,bins2,patches1 = plt.hist(df['electron1'][('Photon1','pt')],bins=50,alpha=0.3,color='blue',cumulative=1,density=False)
n4,bins2,patches1 = plt.hist(df['electron1_pashipt'][('Photon1','pt')],bins=50,alpha=0.3,color='green',cumulative=1,density=False)
```



In [20]:
```
plt.scatter(bins1[:-1],n2/n1,alpha=0.8,color='blue')
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x7f44966cee80>
```



Now lets check corelation between the parameters we are going to use in as input for our DNN

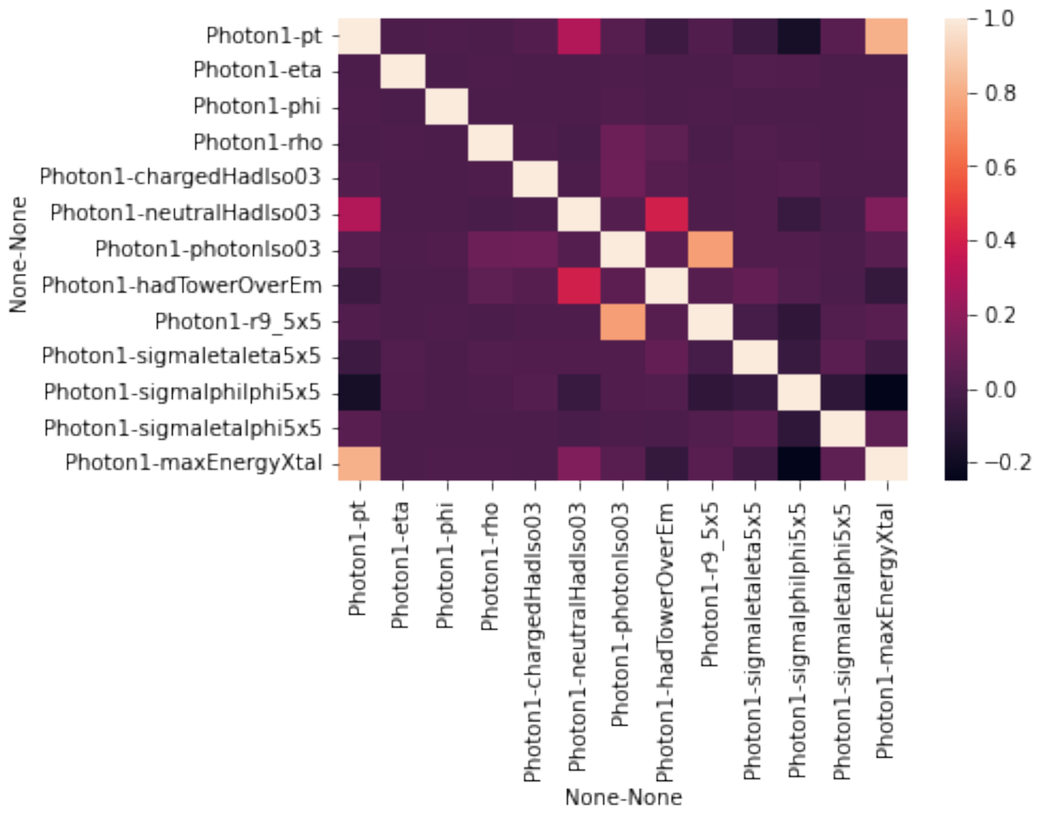
In [21]:
```
features1 = [('Photon1','pt'),('Photon1','eta'),('Photon1','phi'),('Photon1','rho'),('Photon1','chargedHadIso03'),('Photon1','neutralHadIso03'),('Photon1','photonIso03'),(
features2 = [('Photon2','pt'),('Photon2','eta'),('Photon2','phi'),('Photon2','rho'),('Photon2','chargedHadIso03'),('Photon2','neutralHadIso03'),('Photon2','photonIso03'),(
features = [('Photon1','pt'),('Photon1','eta'),('Photon1','phi'),('Photon1','rho'),('Photon1','chargedHadIso03'),('Photon1','neutralHadIso03'),('Photon1','photonIso03'),('
```

In [22]:
```
df['photon1'][('Photon1','iEta')].value_counts()
```

Out[22]:
```
0.0    300766
Name: (Photon1, iEta), dtype: int64
```

In [23]:
```
corr = df['photon1'][features].corr()
```

In [24]:
```
sns.heatmap(corr);
```



Adding new coulmn to extending data frame as "track" whose value is 0 if particle passes to CSEV (passElectronVeto) else 1.

In [25]:
```
def is_track1(row):
    if (row[('Photon1','passElectronVeto')] == 1):
        val = 0
    else:
        val = 1
    return val

def is_track2(row):
    if (row[('Photon2','passElectronVeto')] == 1):
        val = 0
    else:
        val = 1
    return val
df['photon1'][('Photon1','track')] = df['photon1'].apply(is_track1, axis=1)
df['electron1'][('Photon1','track')] = df['electron1'].apply(is_track1, axis=1)
df['photon2'][('Photon2','track')] = df['photon2'].apply(is_track2, axis=1)
df['electron2'][('Photon2','track')] = df['electron2'].apply(is_track2, axis=1)
```

In [26]:
```
# df['electron2'][('Photon2','track')].value_counts()
```

In [27]:
```
# df['photon2'][('Photon2','track')].value_counts()
```

In [28]:
```
df_all1 = pd.concat([df['photon1'], df['electron1']])
df_all2 = pd.concat([df['photon2'], df['electron2']])
```

In [29]:

```python


# df_all1[('Photon1','passElectronVeto')]
```

In [30]:

```python
# df_all1
```

In [31]:

```python
X1 = df_all1[features1].values
Y1 = df_all1['isPhoton']
X2 = df_all2[features2].values
Y2 = df_all2['isPhoton']
X_track=np.concatenate((X1, X2), axis=0)
Y=np.concatenate((Y1, Y2), axis=0)
```

In [32]:

```python

from sklearn.model_selection import train_test_split
X_track_train_val, X_track_test, Y_train_val, Y_test = train_test_split(X_track, Y, test_size=0.20, random_state=7)

from sklearn.preprocessing import StandardScaler
scaler_track = StandardScaler().fit(X_track_train_val)
X_track_train_val = scaler_track.transform(X_track_train_val)
X_track_test = scaler_track.transform(X_track_test)
```

In [33]:

```python
print ("Number of total examples: " + str(X_track.shape[0]))
print ("Number of training examples: " + str(X_track_train_val.shape[0]))
print ("Number of testing examples: " + str(X_track_test.shape[0]))
print ("X_train_val shape: " + str(X_track_train_val.shape))
print ("Y_train_val shape: " + str(Y_train_val.shape))
print ("X_test shape: " + str(X_track_test.shape))
print ("Y_test shape: " + str(Y_test.shape))
```

```
Number of total examples: 880521
Number of training examples: 704416
Number of testing examples: 176105
X_train_val shape: (704416, 14)
Y_train_val shape: (704416,)
X_test shape: (176105, 14)
Y_test shape: (176105,)
```

NN Model with tracker information added for EE region

We are traing this model for 13 + 1(track as a parameter) = 14 for this model

In [34]:

```python
keras.backend.clear_session()
model_EE_track = Sequential()
model_EE_track.add(Dense(300, input_dim=14, activation='relu'))
model_EE_track.add(Dropout(.05))
model_EE_track.add(Dense(250, activation='relu'))
model_EE_track.add(Dropout(.05))
model_EE_track.add(Dense(200, activation='relu'))
model_EE_track.add(Dropout(.05))
model_EE_track.add(Dense(150, activation='relu'))
model_EE_track.add(Dropout(.05))
model_EE_track.add(Dense(100, activation='relu'))
model_EE_track.add(Dropout(.025))
model_EE_track.add(Dense(70, activation='relu'))
model_EE_track.add(Dropout(.01))
model_EE_track.add(Dense(50, activation='relu'))
model_EE_track.add(Dense(25, activation='relu'))
model_EE_track.add(Dense(1, activation='sigmoid'))

# compile the model
model_EE_track.compile(optimizer='nadam', loss='binary_crossentropy', metrics=['accuracy'])
# print the model summary
model_EE_track.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 300)               4500
_________________________________________________________________
dropout (Dropout)            (None, 300)               0
_________________________________________________________________
dense_1 (Dense)              (None, 250)               75250
_________________________________________________________________
dropout_1 (Dropout)          (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 200)               50200
_________________________________________________________________
dropout_2 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_3 (Dense)              (None, 150)               30150
_________________________________________________________________
dropout_3 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               15100
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_5 (Dense)              (None, 70)                7070
_________________________________________________________________
dropout_5 (Dropout)          (None, 70)                0
_________________________________________________________________
dense_6 (Dense)              (None, 50)                3550
_________________________________________________________________
dense_7 (Dense)              (None, 25)                1275
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 26
=================================================================
Total params: 187,121
Trainable params: 187,121
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
2022-04-19 20:47:29.041226: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to u
se the following CPU instructions in performance-critical operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

In [35]:

```python
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_delta=0.0001, min_lr=1e-10, mode='auto')
checkpoint_cb = keras.callbacks.ModelCheckpoint("model_EE_track.h5", save_best_only=True)
early_stopping_cb = keras.callbacks.EarlyStopping(patience=4, restore_best_weights = True)
history=model_EE_track.fit(X_track_train_val, Y_train_val,\
        batch_size=256,\
        epochs=50,\
        validation_split=.20,\
        callbacks=[reduce_lr, checkpoint_cb, early_stopping_cb],\
        verbose=1, shuffle=True, initial_epoch=0
        )
```

```
Epoch 1/50

2022-04-19 20:47:29.255252: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2)
2022-04-19 20:47:29.256005: I tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency: 2099945000 Hz

2202/2202 [==============================] - 19s 8ms/step - loss: 0.2650 - accuracy: 0.8953 - val_loss: 0.2428 - val_accuracy: 0.9026
Epoch 2/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2363 - accuracy: 0.9040 - val_loss: 0.2345 - val_accuracy: 0.9035
Epoch 3/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2317 - accuracy: 0.9055 - val_loss: 0.2376 - val_accuracy: 0.9058
Epoch 4/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2291 - accuracy: 0.9065 - val_loss: 0.2283 - val_accuracy: 0.9070
Epoch 5/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2280 - accuracy: 0.9069 - val_loss: 0.2283 - val_accuracy: 0.9065
Epoch 6/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2270 - accuracy: 0.9068 - val_loss: 0.2276 - val_accuracy: 0.9071
Epoch 7/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2270 - accuracy: 0.9075 - val_loss: 0.2272 - val_accuracy: 0.9070
Epoch 8/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2246 - accuracy: 0.9083 - val_loss: 0.2259 - val_accuracy: 0.9080
Epoch 9/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2245 - accuracy: 0.9082 - val_loss: 0.2265 - val_accuracy: 0.9079
Epoch 10/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2242 - accuracy: 0.9085 - val_loss: 0.2254 - val_accuracy: 0.9077
Epoch 11/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2233 - accuracy: 0.9085 - val_loss: 0.2288 - val_accuracy: 0.9073
Epoch 12/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2217 - accuracy: 0.9092 - val_loss: 0.2268 - val_accuracy: 0.9071
Epoch 13/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2190 - accuracy: 0.9100 - val_loss: 0.2230 - val_accuracy: 0.9090
Epoch 14/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2172 - accuracy: 0.9108 - val_loss: 0.2229 - val_accuracy: 0.9090
Epoch 15/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2177 - accuracy: 0.9107 - val_loss: 0.2234 - val_accuracy: 0.9089
Epoch 16/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2173 - accuracy: 0.9110 - val_loss: 0.2227 - val_accuracy: 0.9090
Epoch 17/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2156 - accuracy: 0.9116 - val_loss: 0.2227 - val_accuracy: 0.9091
Epoch 18/50
```

```
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2151 - accuracy: 0.9118 - val_loss: 0.2225 - val_accuracy: 0.9093
Epoch 19/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2161 - accuracy: 0.9113 - val_loss: 0.2227 - val_accuracy: 0.9091
Epoch 20/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2149 - accuracy: 0.9118 - val_loss: 0.2226 - val_accuracy: 0.9091
Epoch 21/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2149 - accuracy: 0.9121 - val_loss: 0.2225 - val_accuracy: 0.9093
Epoch 22/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.2158 - accuracy: 0.9113 - val_loss: 0.2226 - val_accuracy: 0.9092
```

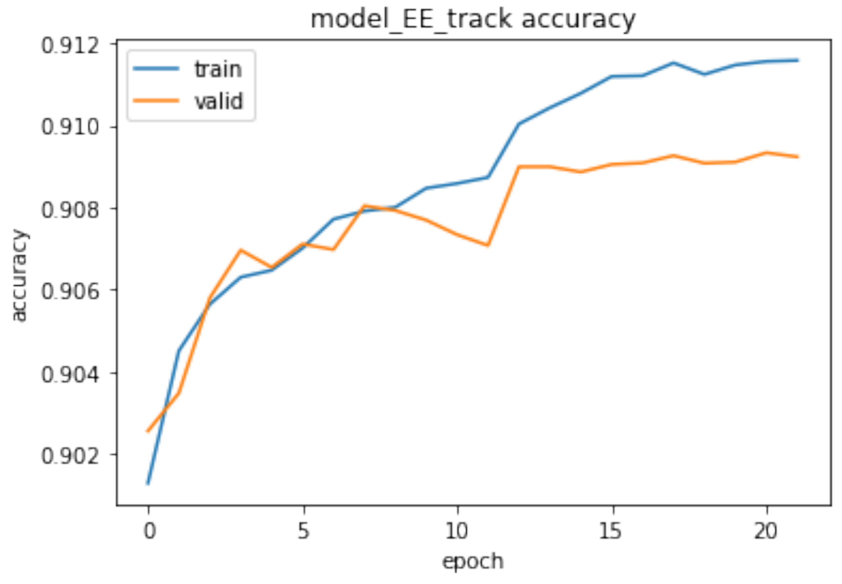
In [36]:
```
model_EE_track = keras.models.load_model("model_EE_track.h5")
```

In [37]:
```
model_EE_track.evaluate(X_track_test, Y_test)
```

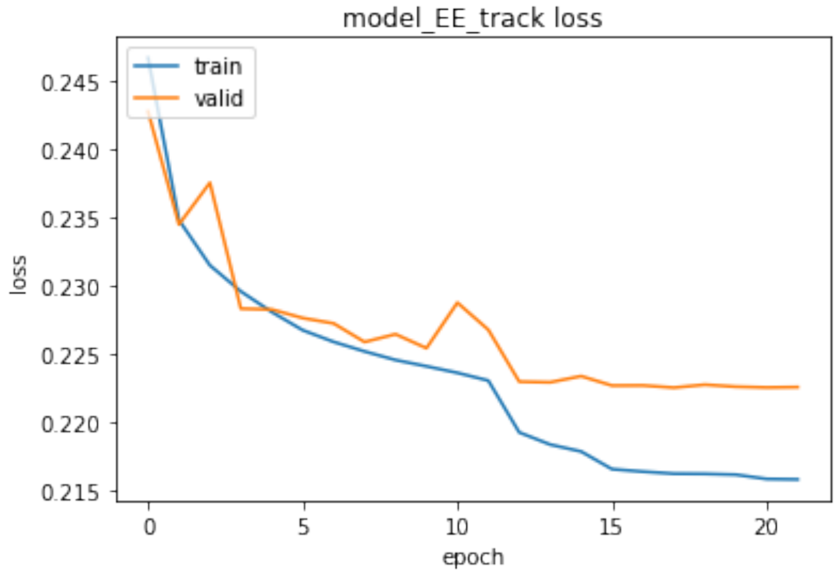
```
5504/5504 [==============================] - 6s 1ms/step - loss: 0.2196 - accuracy: 0.9105
```

Out[37]:
```
[0.2196447253227234, 0.9104511737823486]
```

In [38]:
```
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model_EE_track accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```



In [39]:
```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model_EE_track loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```



In [40]:
```
y_track_pred=model_EE_track.predict(X_track_test)
```

In [41]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test, y_track_pred.round())
```

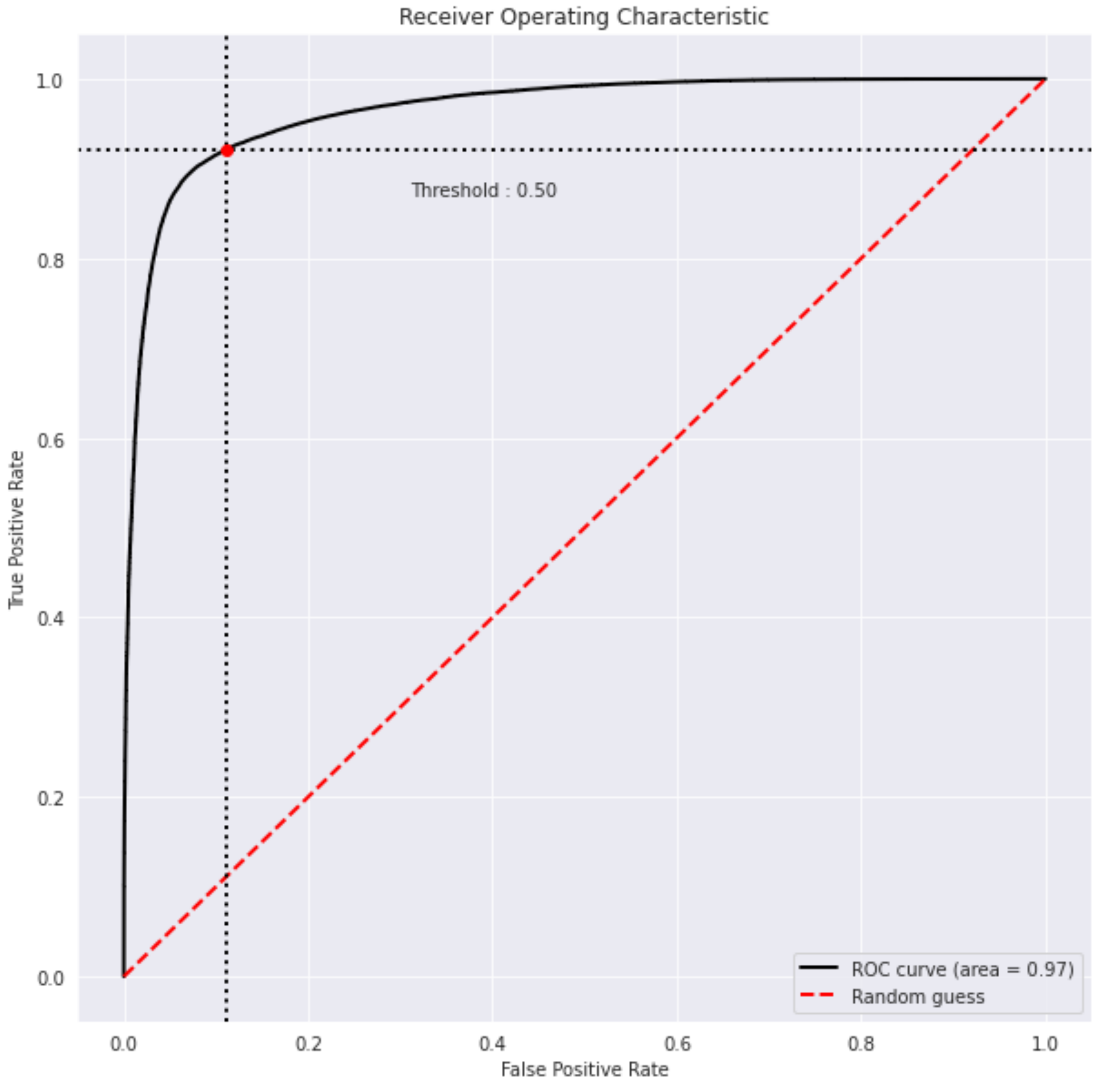
Out[41]:
```
array([[ 51904,   6481],
       [  9289, 108431]])
```

In [42]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

from plot_metric.functions import BinaryClassification


# Visualisation with plot_metric
bc = BinaryClassification(Y_test, y_track_pred, labels=["Class 1", "Class 2"])

# Figures
plt.figure(figsize=(10,10))
bc.plot_roc_curve()
plt.show()
```



Removing tracking information from input data

In [43]:
```
X_notrack_train_val = X_track_train_val[:,0:13]
X_notrack_test = X_track_test[:,0:13]
```

NN Model without tracker information for EE region

We are only training with 13 parameter for this model

In [44]:
```
keras.backend.clear_session()
model_EE_notrack = Sequential()
model_EE_notrack.add(Dense(300, input_dim=13, activation='relu'))
model_EE_notrack.add(Dropout(.05))
model_EE_notrack.add(Dense(250, activation='relu'))
model_EE_notrack.add(Dropout(.05))
model_EE_notrack.add(Dense(200, activation='relu'))
model_EE_notrack.add(Dropout(.05))
model_EE_notrack.add(Dense(150, activation='relu'))
model_EE_notrack.add(Dropout(.05))
model_EE_notrack.add(Dense(100, activation='relu'))
model_EE_notrack.add(Dropout(.025))
model_EE_notrack.add(Dense(70, activation='relu'))
model_EE_notrack.add(Dropout(.01))
model_EE_notrack.add(Dense(50, activation='relu'))
model_EE_notrack.add(Dense(25, activation='relu'))
model_EE_notrack.add(Dense(1, activation='sigmoid'))

# compile the model
model_EE_notrack.compile(optimizer='nadam', loss='binary_crossentropy', metrics=['accuracy'])
# print the model summary
model_EE_notrack.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 300)               4200
_________________________________________________________________
dropout (Dropout)            (None, 300)               0
_________________________________________________________________
dense_1 (Dense)              (None, 250)               75250
_________________________________________________________________
dropout_1 (Dropout)          (None, 250)               0
_________________________________________________________________
dense_2 (Dense)              (None, 200)               50200
_________________________________________________________________
dropout_2 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_3 (Dense)              (None, 150)               30150
_________________________________________________________________
dropout_3 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_4 (Dense)              (None, 100)               15100
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_5 (Dense)              (None, 70)                7070
_________________________________________________________________
dropout_5 (Dropout)          (None, 70)                0
_________________________________________________________________
```

```
dense_6 (Dense)              (None, 50)                3550      
_________________________________________________________________
dense_7 (Dense)              (None, 25)                1275      
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 26        
=================================================================
Total params: 186,821
Trainable params: 186,821
Non-trainable params: 0
_________________________________________________________________
```

In [45]:

```python
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_delta=0.0001, min_lr=1e-10, mode='auto')
checkpoint_cb = keras.callbacks.ModelCheckpoint("model_EE_notrack.h5", save_best_only=True)
early_stopping_cb = keras.callbacks.EarlyStopping(patience=4, restore_best_weights = True)
history=model_EE_notrack.fit(X_notrack_train_val, Y_train_val,\
        batch_size=256,\
        epochs=50,\
        validation_split=.20,\
        callbacks=[reduce_lr, checkpoint_cb, early_stopping_cb],\
        verbose=1, shuffle=True, initial_epoch=0
        )
```

```
Epoch 1/50
2202/2202 [==============================] - 18s 7ms/step - loss: 0.4780 - accuracy: 0.7761 - val_loss: 0.4347 - val_accuracy: 0.8021
Epoch 2/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4343 - accuracy: 0.8036 - val_loss: 0.4300 - val_accuracy: 0.8039
Epoch 3/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4303 - accuracy: 0.8045 - val_loss: 0.4269 - val_accuracy: 0.8069
Epoch 4/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4273 - accuracy: 0.8062 - val_loss: 0.4254 - val_accuracy: 0.8069
Epoch 5/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4261 - accuracy: 0.8068 - val_loss: 0.4254 - val_accuracy: 0.8064
Epoch 6/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4245 - accuracy: 0.8076 - val_loss: 0.4266 - val_accuracy: 0.8071
Epoch 7/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4216 - accuracy: 0.8076 - val_loss: 0.4205 - val_accuracy: 0.8089
Epoch 8/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4168 - accuracy: 0.8105 - val_loss: 0.4198 - val_accuracy: 0.8088
Epoch 9/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4184 - accuracy: 0.8090 - val_loss: 0.4196 - val_accuracy: 0.8089
Epoch 10/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4159 - accuracy: 0.8110 - val_loss: 0.4196 - val_accuracy: 0.8087
Epoch 11/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4160 - accuracy: 0.8109 - val_loss: 0.4195 - val_accuracy: 0.8084
Epoch 12/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4151 - accuracy: 0.8108 - val_loss: 0.4186 - val_accuracy: 0.8091
Epoch 13/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4123 - accuracy: 0.8123 - val_loss: 0.4186 - val_accuracy: 0.8089
Epoch 14/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4149 - accuracy: 0.8106 - val_loss: 0.4187 - val_accuracy: 0.8088
Epoch 15/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4141 - accuracy: 0.8111 - val_loss: 0.4185 - val_accuracy: 0.8090
Epoch 16/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4137 - accuracy: 0.8116 - val_loss: 0.4185 - val_accuracy: 0.8091
Epoch 17/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4121 - accuracy: 0.8131 - val_loss: 0.4185 - val_accuracy: 0.8090
Epoch 18/50
```

```
2202/2202 [==============================] - 17s 8ms/step - loss: 0.4136 - accuracy: 0.8117 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 19/50
2202/2202 [==============================] - 18s 8ms/step - loss: 0.4135 - accuracy: 0.8116 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 20/50
2202/2202 [==============================] - 17s 8ms/step - loss: 0.4141 - accuracy: 0.8109 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 21/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4128 - accuracy: 0.8118 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 22/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4125 - accuracy: 0.8126 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 23/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4126 - accuracy: 0.8125 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 24/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4120 - accuracy: 0.8130 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 25/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4129 - accuracy: 0.8121 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 26/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4137 - accuracy: 0.8119 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 27/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4128 - accuracy: 0.8119 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 28/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4123 - accuracy: 0.8132 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 29/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4150 - accuracy: 0.8104 - val_loss: 0.4185 - val_accuracy: 0.8089
Epoch 30/50
2202/2202 [==============================] - 16s 7ms/step - loss: 0.4137 - accuracy: 0.8117 - val_loss: 0.4185 - val_accuracy: 0.8089
```

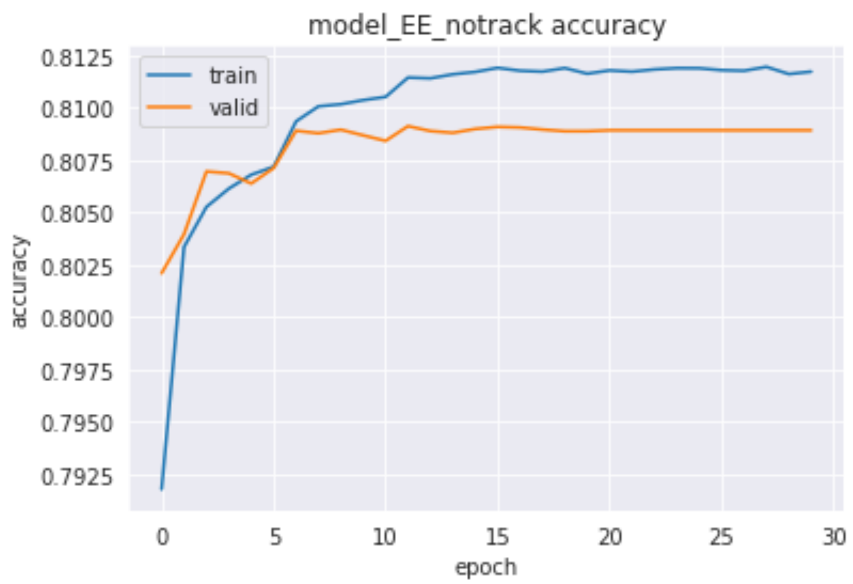
In [47]:
```
model_EE_notrack = keras.models.load_model("model_EE_notrack.h5")
```

In [48]:
```
model_EE_notrack.evaluate(X_notrack_test, Y_test)
```

```
5504/5504 [==============================] - 6s 1ms/step - loss: 0.4167 - accuracy: 0.8102
```

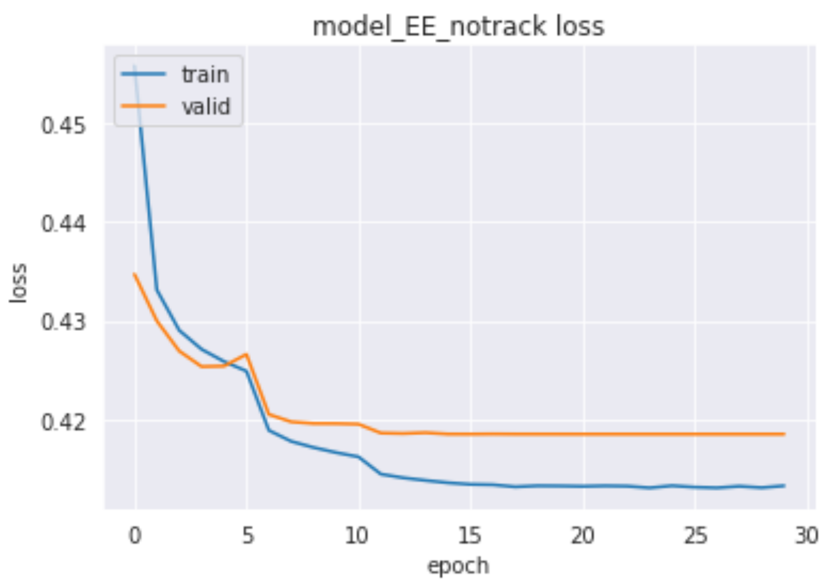
Out[48]:
```
[0.4166813790798187, 0.8101757764816284]
```

In [49]:
```python
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model_EE_notrack accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()
```



In [50]:
```python
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model_EE_notrack loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')

plt.show()
```



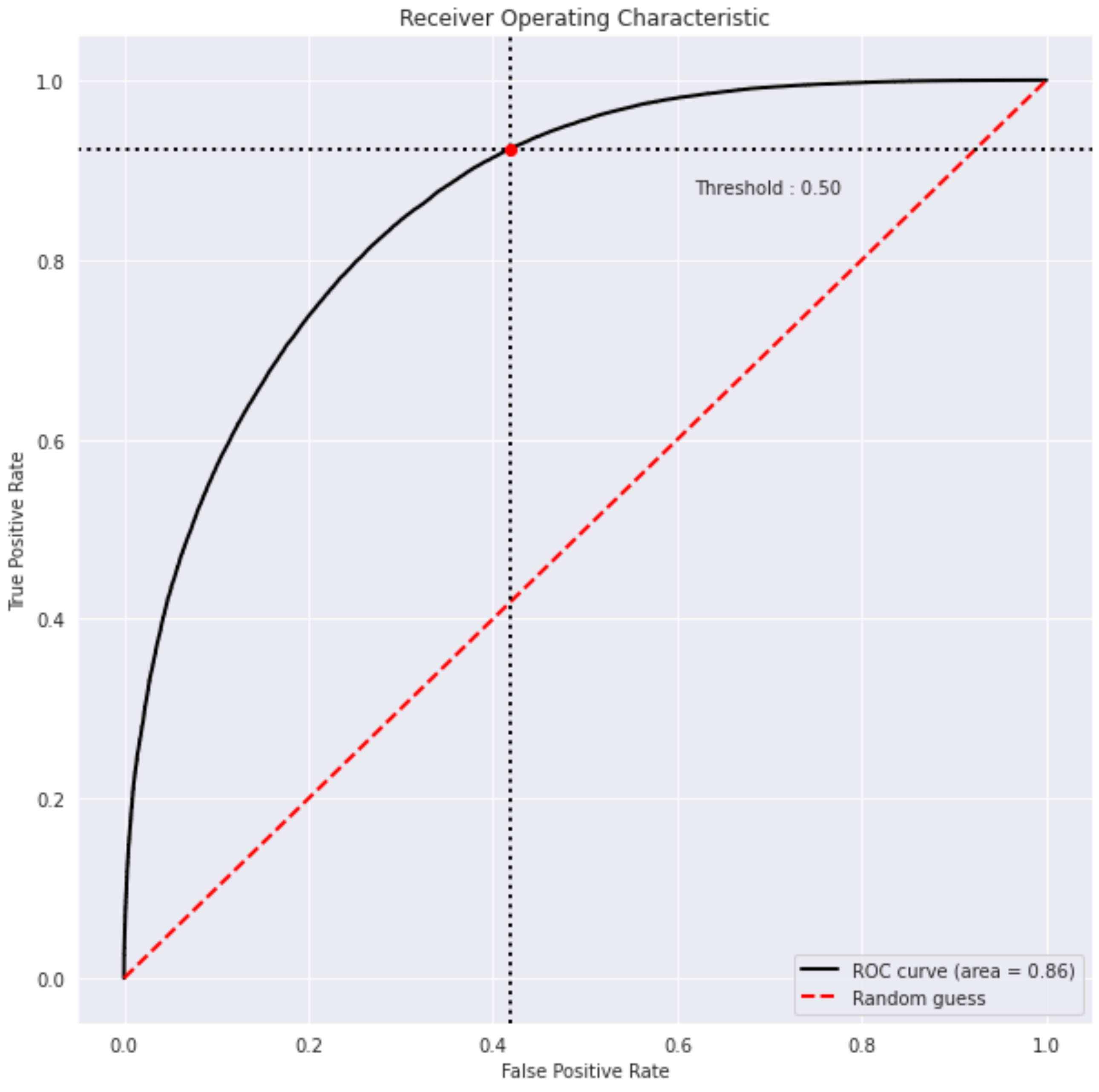
In [51]:
```python
y_notrack_pred=model_EE_notrack.predict(X_notrack_test)
```

In [52]:
```
confusion_matrix(Y_test, y_notrack_pred.round())
```

Out[52]:
```
array([[ 33942,  24443],
       [  8986, 108734]])
```

In [53]:
```
# Visualisation with plot_metric
bc = BinaryClassification(Y_test, y_notrack_pred, labels=["Class 1", "Class 2"])

# Figures
plt.figure(figsize=(10,10))
bc.plot_roc_curve()
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: