In [4]:

```
import numpy as np
import pandas as pd
import seaborn as sns

from sklearn.calibration import calibration_curve
from sklearn.model_selection import cross_val_score,cross_val_predict, train_test_split
from sklearn.preprocessing import OneHotEncoder,StandardScaler,PowerTransformer,LabelEncoder
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import accuracy_score,classification_report, recall_score,confusion_matrix, roc_auc_score, precision_score, f
from sklearn.calibration import CalibratedClassifierCV

import matplotlib.pyplot as plt

from scipy.sparse import csr_matrix

import scikitplot as skplt

import optuna
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier


#importing plotly and cufflinks in offline mode
import cufflinks as cf
import plotly.offline
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)


import plotly
import plotly.express as px
import plotly.graph_objs as go
import plotly.offline as py
from plotly.offline import iplot
from plotly.subplots import make_subplots
import plotly.figure_factory as ff

import shap

import missingno as msno
```

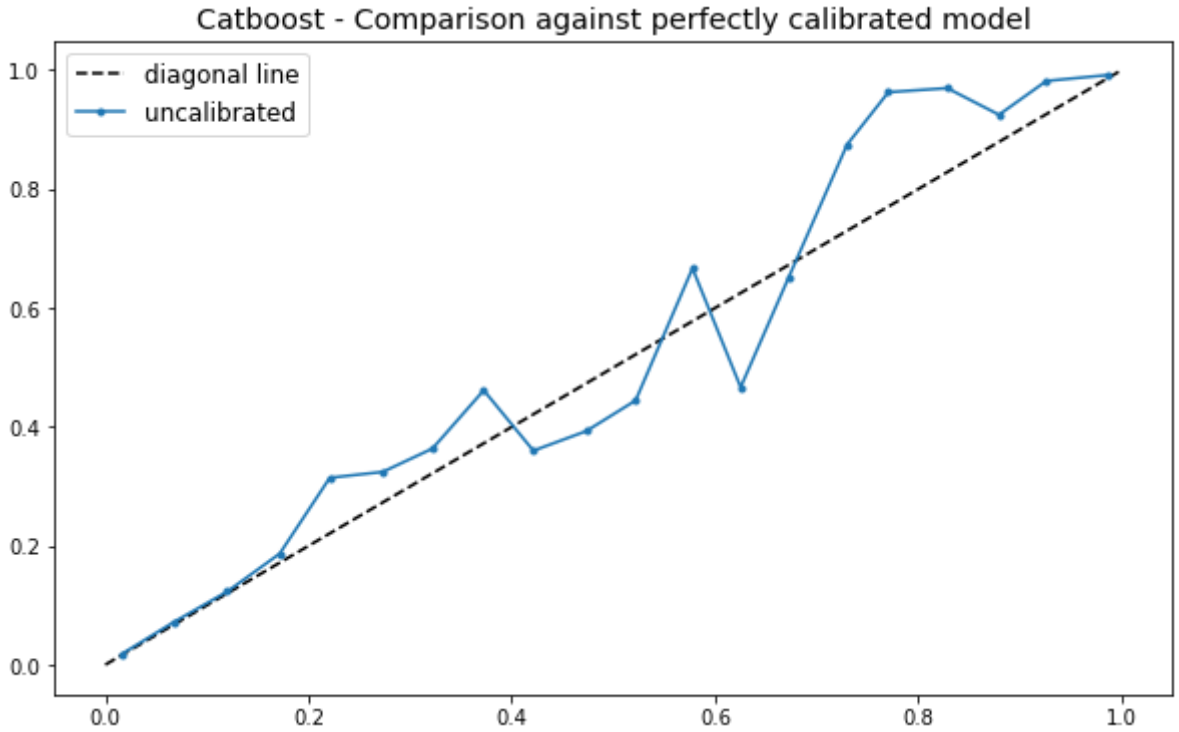
```
import warnings
warnings.filterwarnings("ignore")
```

In [5]:
```
df = pd.read_csv(r'C:\Users\esp1_\OneDrive\Skrivebord\thesis/data_v1.9.csv', low_memory = False)

categorical = df.select_dtypes('object').columns
categorical = categorical.tolist()
df[categorical] = df[categorical].fillna('missing')

df = df.drop(['contact_id', 'Inscrito', 'Admitido', 'Useless', 'Contacto', 'Asistencia', 'Cita', 'no contactado', 'original_source
```

In [6]:
```
df_dummies = pd.get_dummies(df)
df1_dummies_1 = df_dummies.loc[df_dummies['Ciclo Generación_2021-3'] == 1]
df1_dummies_2 = df_dummies.loc[df_dummies['Ciclo Generación_2022-1'] == 1]
df1 = pd.concat([df1_dummies_1,df1_dummies_2])
df2_dummies_1 = df_dummies.loc[df_dummies['Ciclo Generación_2022-2'] == 1]
df2_dummies_2 = df_dummies.loc[df_dummies['Ciclo Generación_2022-3'] == 1]
df2 = pd.concat([df2_dummies_1,df2_dummies_2])

df1 = df1.drop(['Ciclo Generación_2021-3', 'Ciclo Generación_2022-1', 'Ciclo Generación_2022-2', 'Ciclo Generación_2022-3'], axis=
df2 = df2.drop(['Ciclo Generación_2021-3', 'Ciclo Generación_2022-1', 'Ciclo Generación_2022-2', 'Ciclo Generación_2022-3'], axis=
```

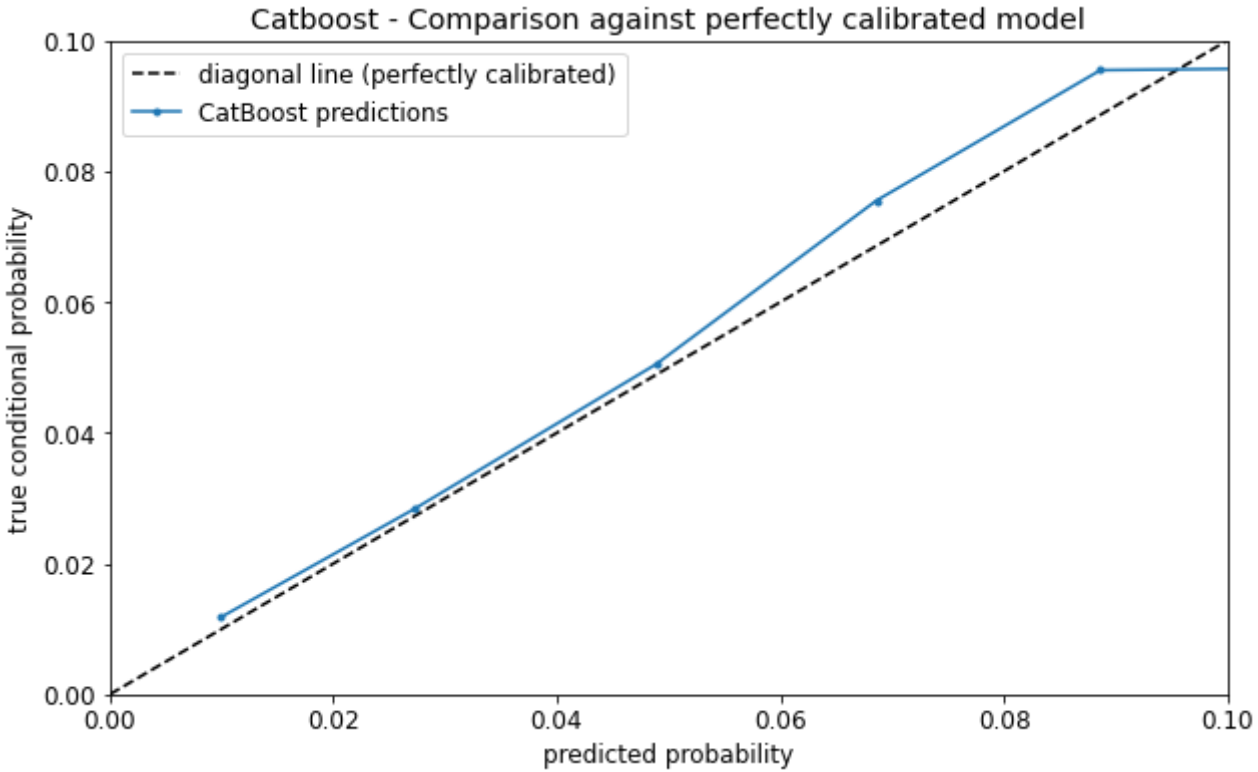
In [7]:
```
X = df1.drop('enrolled', axis=1)
y = df1['enrolled']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


# predict uncalibrated probabilities
def uncalibrated(X_train, X_test, y_train):

    catboost = CatBoostClassifier(verbose=False,random_state=0, task_type='GPU') # , scale_pos_weight=9
    catboost.fit(X_train, y_train)
    return catboost.predict_proba(X_test)[:, 1]
'''
# predict calibrated probabilities
def calibrated(X_train, X_test, y_train):

    catboost = CatBoostClassifier(verbose=False,random_state=0, task_type='GPU') # , scale_pos_weight=9
```

```
    calibrated = CalibratedClassifierCV(catboost, method='sigmoid', cv=5)
    calibrated.fit(X_train, y_train)
    return calibrated.predict_proba(X_test)[:, 1]
'''

# uncalibrated predictions
yhat_uncalibrated = uncalibrated(X_train, X_test, y_train)
# calibrated predictions
# yhat_calibrated = calibrated(X_train, X_test, y_train)
# reliability diagrams
fop_uncalibrated, mpv_uncalibrated = calibration_curve(y_test, yhat_uncalibrated, n_bins=20, normalize=True)
# fop_calibrated, mpv_calibrated = calibration_curve(y_test, yhat_calibrated, n_bins=20)
# plot perfectly calibrated
plt.subplots(1, figsize=(10,6))
plt.rcParams['font.size'] = '12'
plt.title('Catboost - Comparison against perfectly calibrated model')
plt.plot([0, 1], [0, 1], linestyle='--', color='black')
# plot model reliabilities
plt.plot(mpv_uncalibrated, fop_uncalibrated, marker='.')
# plt.plot(mpv_calibrated, fop_calibrated, marker='.')
plt.legend(['diagonal line','uncalibrated', 'calibrated'])
plt.show()
```



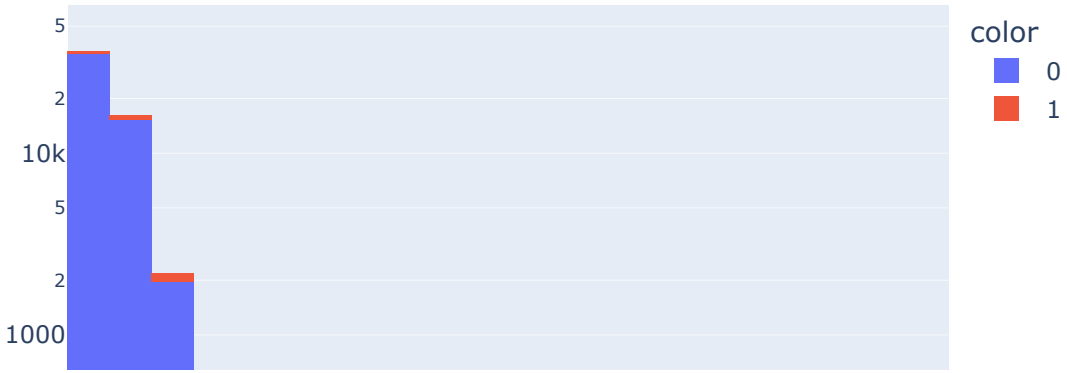
In [9]:

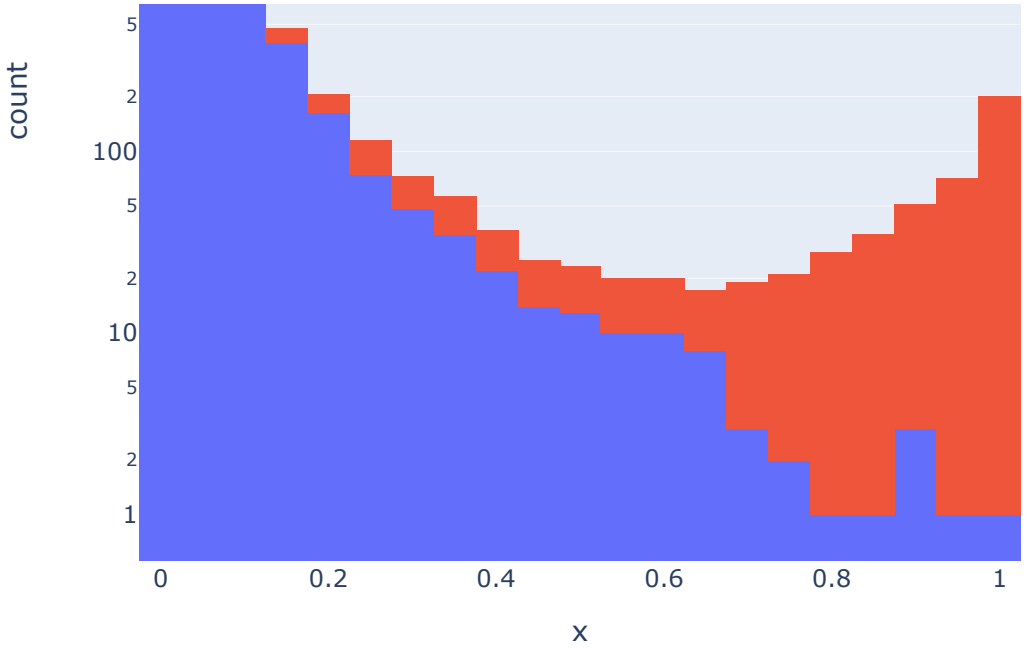
```
# reliability diagrams
fop_uncalibrated, mpv_uncalibrated = calibration_curve(y_test, yhat_uncalibrated, n_bins=50, normalize=True)
#fop_calibrated, mpv_calibrated = calibration_curve(y_test, yhat_calibrated, n_bins=50)
# plot perfectly calibrated
plt.subplots(1, figsize=(10,6))
plt.rcParams['font.size'] = '14'
plt.xlim([0, 0.1])
plt.ylim([0, 0.1])
plt.xlabel("predicted probability")
plt.ylabel("true conditional probability")
plt.rcParams['font.size'] = '12'
plt.title('Catboost - Comparison against perfectly calibrated model')
plt.plot([0, 1], [0, 1], linestyle='--', color='black')
# plot model reliabilities
plt.plot(mpv_uncalibrated, fop_uncalibrated, marker='.')
#plt.plot(mpv_calibrated, fop_calibrated, marker='.')
plt.legend(['diagonal line (perfectly calibrated)','CatBoost predictions', 'calibrated'])
plt.show()
```



In [80]:

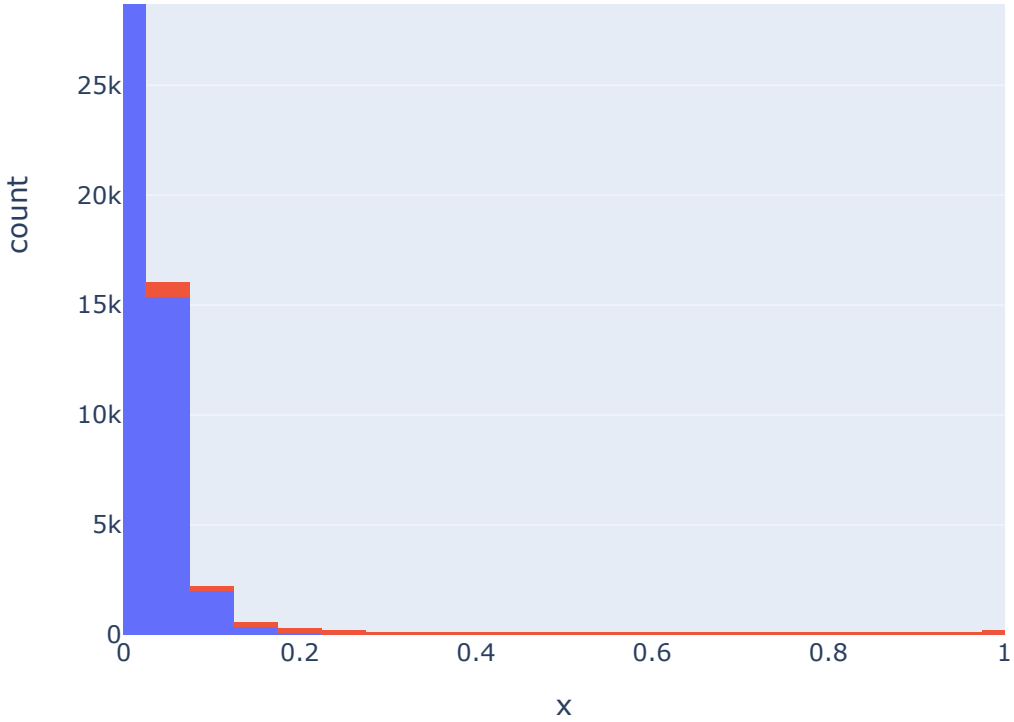
```
fig = px.histogram(df, x=yhat_uncalibrated, color=y_test,width=600, height=600, nbins=20, log_y = True)
fig.show()
```





In [66]:
```
#fig = px.histogram(df, x=yhat_calibrated, color=y_test,width=600, height=600, nbins=40, range_x=[0,1], title = 'Histogram of Cali
#fig.show()
```

In [67]:
```
fig = px.histogram(df, x=yhat_uncalibrated, color=y_test,width=600, height=600, nbins=40, range_x=[0,1], title = 'Histogram of Cat
fig.show()
```



In [68]:

```
accuracy= []
recall =[]
roc_auc= []
precision = []
model_names =[]

# categorical_features_indices = np.where(X_train.dtypes != np.float)[0]

catboost = CatBoostClassifier(verbose=False,random_state=0, task_type='GPU') #, scale_pos_weight=9

catboost.fit(X_train, y_train, eval_set=(X_test, y_test)) # cat_features=categorical_features_indices,

y_pred1_shap = catboost.predict(X_test)
y_pred_shap = catboost.predict_proba(X_test)

accuracy.append(round(accuracy_score(y_test, y_pred1_shap),4))
recall.append(round(recall_score(y_test, y_pred1_shap),4))
```

```
roc_auc.append(round(roc_auc_score(y_test, y_pred_shap[:,1]),4))
precision.append(round(precision_score(y_test, y_pred1_shap),4))

model_names = ['Catboost']
result_Catbooste = pd.DataFrame({'Accuracy':accuracy,'Recall':recall, 'Roc_Auc':roc_auc, 'Precision':precision}, index=model_names
result_Catbooste
```

Out[68]:

	Accuracy	Recall	Roc_Auc	Precision
Catboost	0.9717	0.2294	0.8164	0.9014

In [69]:

```
Leads_costing = 13000

Talent    = 800000
Telco     = 44132
Software  = 35000
Marketing = 1550000
Hardware  = 75000
Total     = Talent + Telco + Software + Marketing + Hardware

Cost_p_lead = (Total/Leads_costing)*-1

Revenue   = 33080
```

In [70]:

```
column_names = ['profit', 'threshold']
profit_df = pd.DataFrame(columns = column_names)

for i in range(1000):

    y_pred_opt_1 = (catboost.predict_proba(X_test)[:, 1] > (i+1)/1000).astype('float')
    x = confusion_matrix(y_test, y_pred_opt_1)
    FP = x[0,1]*-192.63
    TP = x[1,1]*33080
    profit = TP+FP
    profit_df.loc[i, 'profit'] = profit
    profit_df.loc[i, 'threshold'] = (i+1)/1000
```

In [71]:

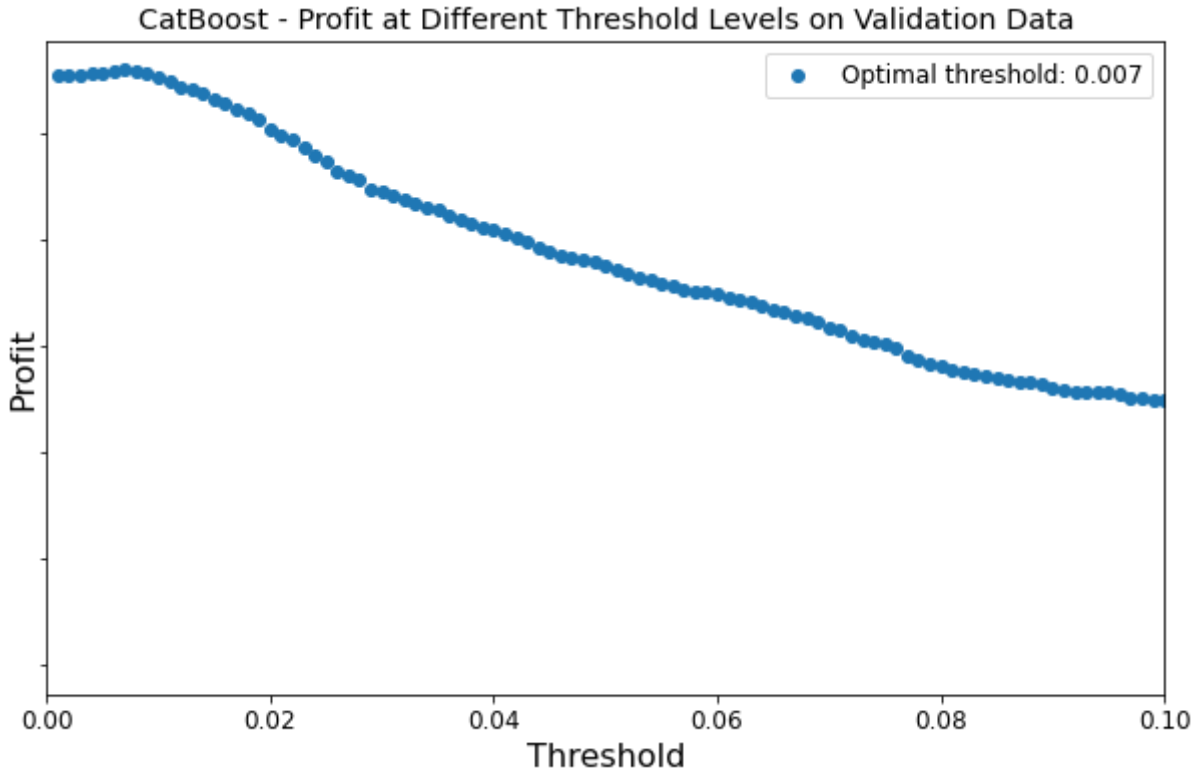
```
profit_df['profit'] = pd.to_numeric(profit_df['profit'])
profit_df['threshold'] = pd.to_numeric(profit_df['threshold'])
```

```
print(profit_df[['profit']].idxmax())
print(profit_df[['profit']].max())
```

```
profit    6
dtype: int64
profit    55923409.76
dtype: float64
```

In [72]:

```
plt.subplots(1, figsize=(10,6))
plt.rcParams['font.size'] = '12'
plt.title('CatBoost - Profit at Different Threshold Levels on Validation Data')
plt.scatter(profit_df['threshold'], profit_df['profit'])
plt.xlabel("Threshold", fontsize=16)
plt.tick_params(labelleft=False)
plt.ylabel("Profit", fontsize=16)
profit_threshold = profit_df.loc[6,:] # set threshold location found in prev cell
plt.xlim([0.0, .1])
p1 = 'Optimal threshold: ' + str(profit_threshold[1])
plt.legend([p1])
plt.show()
```



In [73]:
```
X_ho = df2.drop('enrolled', axis=1)
y_ho = df2['enrolled']
```

In [74]:
```
# Theoretical threshold
threshold = 0.0058
y_pred = (catboost.predict_proba(X_ho)[:, 1] > threshold).astype('float')
cf_matrix = confusion_matrix(y_ho, y_pred)
print(cf_matrix)
```

```
[[ 1357 58247]
 [    0  3002]]
```

In [75]:
```
FP_Cost = round(cf_matrix[0,1] * Cost_p_lead, 2)
TP_Rev  = round(cf_matrix[1,1] * Revenue, 2)
TN_Rev  = round(cf_matrix[0,0] * 0,2)
```

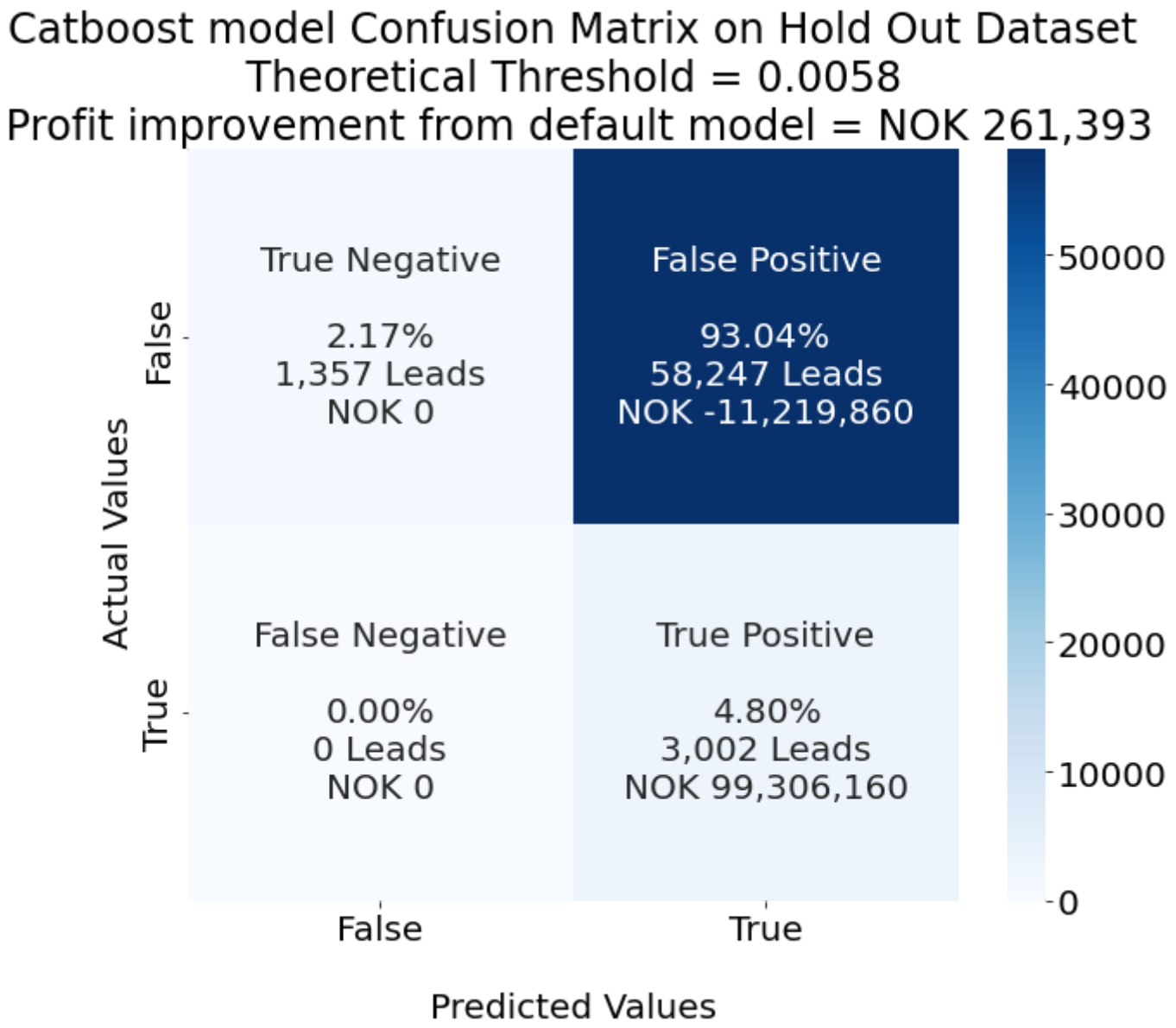
```
FN_Cost = round(cf_matrix[1,0] * 0,2)
profit_matrix = [TN_Rev,FP_Cost, FN_Cost,TP_Rev]

print ("Total costs   = " , Total)
print ("Cost per lead = ", "{:.2f}".format(Cost_p_lead))
print("")
print("The profit_matrix contains: ")
print(profit_matrix)
print("")
print("FP cost = NOK {:.0f}".format(FP_Cost))
print("TP revenue NOK {:.0f}".format(TP_Rev))
print("TN revenue NOK {:.0f}".format(TN_Rev))
print("FN cost NOK {:.0f}" .format(FN_Cost))

Default = 87824907
Profit = FP_Cost+TP_Rev+TN_Rev+FN_Cost -Default
print("The default profit is NOK {:,.0f}".format(Default))
print("The profit over default is NOK {:,.0f}".format(Profit))
```

```
Total costs   =  2504132
Cost per lead =  -192.63

The profit_matrix contains:
[0, -11219859.74, 0, 99306160]

FP cost = NOK -11219860
TP revenue NOK 99306160
TN revenue NOK 0
FN cost NOK 0
The default profit is NOK 87,824,907
The profit over default is NOK 261,393
```

In [76]:

```
group_names = ['True Negative','False Positive','False Negative','True Positive']

group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten()/np.sum(cf_matrix)]

group_counts = ["{0:,.0f} Leads".format(value) for value in
                cf_matrix.flatten()]

profit_each = ["NOK {0:,.0f}".format(value) for value in
                profit_matrix]
```

```
labels = [f"{v1}\n\n{v2}\n{v3}\n{v4}" for v1, v2, v3, v4 in
          zip(group_names,group_percentages,group_counts,profit_each)]

labels = np.asarray(labels).reshape(2,2)

fig, ax = plt.subplots(figsize=(10, 8))
plt.rcParams['font.size'] = '20'

ax = sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues')

#plt.title("Histograms for {0:.2f}".format(df.columns[i]))

ax.set_title('Catboost model Confusion Matrix on Hold Out Dataset\n Theoretical Threshold = 0.0058 \n Profit improvement from defa
ax.set_xlabel('\nPredicted Values', fontsize=20)
ax.set_ylabel('Actual Values ', fontsize=20);

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'], fontsize=20)
ax.yaxis.set_ticklabels(['False','True'], fontsize=20)

## Display the visualization of the Confusion Matrix.
plt.show()
```



In [77]:

```
# Optimal Threshold
threshold = 0.007
y_pred = (catboost.predict_proba(X_ho)[:, 1] > threshold).astype('float')
cf_matrix = confusion_matrix(y_ho, y_pred)
print(cf_matrix)
```

```
[[ 2677 56927]
 [    6  2996]]
```

In [78]:

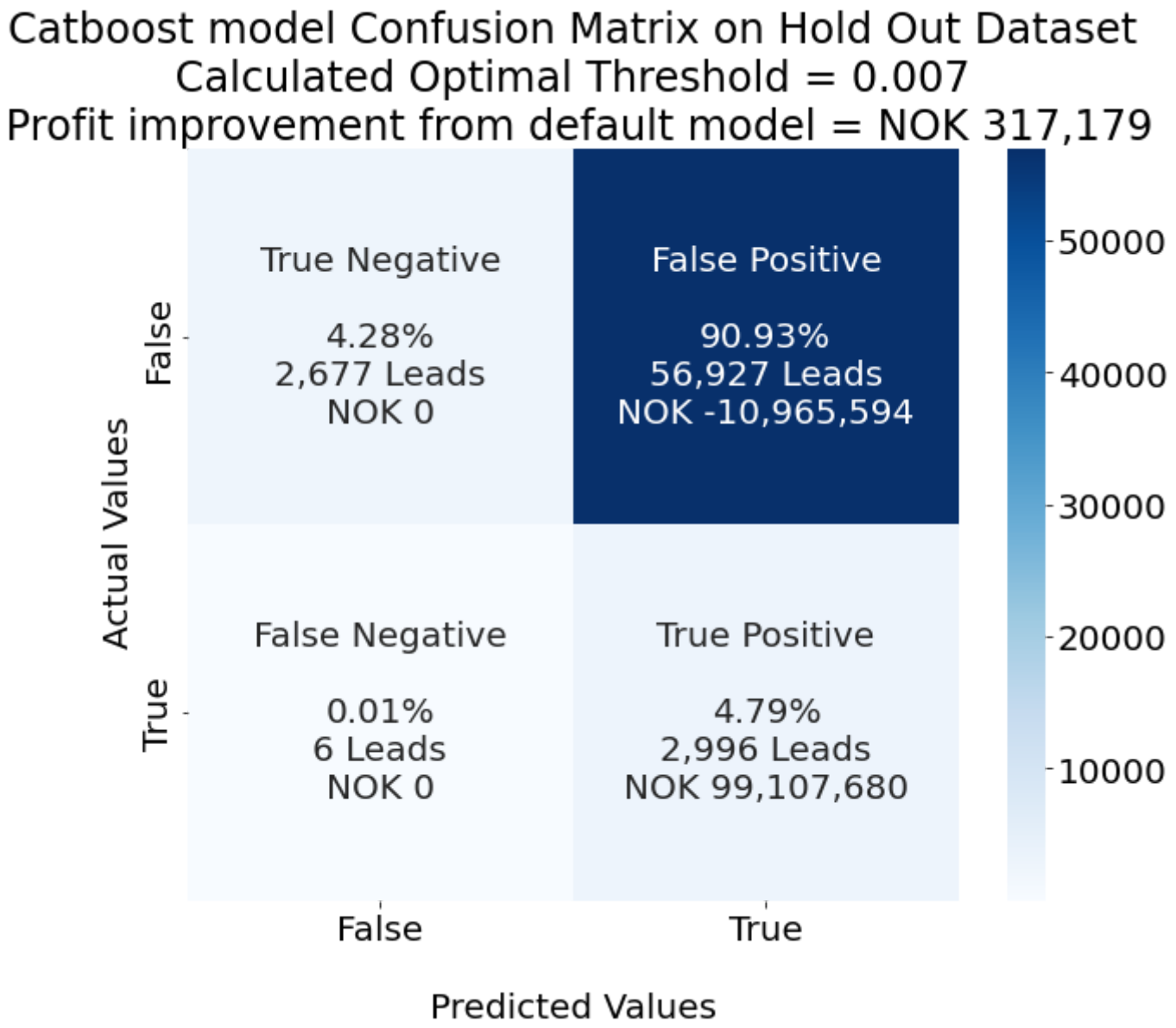
```
FP_Cost = round(cf_matrix[0,1] * Cost_p_lead, 2)
TP_Rev  = round(cf_matrix[1,1] * Revenue, 2)
TN_Rev  = round(cf_matrix[0,0] * 0,2)
FN_Cost = round(cf_matrix[1,0] * 0,2)
profit_matrix = [TN_Rev,FP_Cost, FN_Cost,TP_Rev]

print ("Total costs   = " , Total)
print ("Cost per lead = ", "{:.2f}".format(Cost_p_lead))
print("")
print("The profit_matrix contains: ")
print(profit_matrix)
print("")
print("FP cost = NOK {:.0f}".format(FP_Cost))
print("TP revenue NOK {:.0f}".format(TP_Rev))
print("TN revenue NOK {:.0f}".format(TN_Rev))
print("FN cost NOK {:.0f}" .format(FN_Cost))

Default = 87824907
Profit = FP_Cost+TP_Rev+TN_Rev+FN_Cost - Default
print("The total profit is NOK {:,.0f}".format(Profit))
```

```
Total costs   =  2504132
Cost per lead =  -192.63

The profit_matrix contains:
[0, -10965594.03, 0, 99107680]

FP cost = NOK -10965594
TP revenue NOK 99107680
TN revenue NOK 0
FN cost NOK 0
The total profit is NOK 317,179
```

In [79]:

```
group_names = ['True Negative','False Positive','False Negative','True Positive']

group_percentages = ["{0:.2%}".format(value) for value in
                     cf_matrix.flatten()/np.sum(cf_matrix)]

group_counts = ["{0:,.0f} Leads".format(value) for value in
```

```
                cf_matrix.flatten()]

profit_each = ["NOK {0:,.0f}".format(value) for value in
                profit_matrix]

labels = [f"{v1}\n\n{v2}\n{v3}\n{v4}" for v1, v2, v3, v4 in
          zip(group_names,group_percentages,group_counts,profit_each)]

labels = np.asarray(labels).reshape(2,2)

fig, ax = plt.subplots(figsize=(10, 8))
plt.rcParams['font.size'] = '20'

ax = sns.heatmap(cf_matrix, annot=labels, fmt='', cmap='Blues')

#plt.title("Histograms for {0:.2f}".format(df.columns[i]))

ax.set_title('Catboost model Confusion Matrix on Hold Out Dataset\n Calculated Optimal Threshold = 0.007 \n Profit improvement fro
ax.set_xlabel('\nPredicted Values', fontsize=20)
ax.set_ylabel('Actual Values ', fontsize=20);

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'], fontsize=20)
ax.yaxis.set_ticklabels(['False','True'], fontsize=20)

## Display the visualization of the Confusion Matrix.
plt.show()
```


Catboost model Confusion Matrix on Hold Out Dataset
Calculated Optimal Threshold = 0.007
Profit improvement from default model = NOK 317,179
True Negative
4.28%
2,677 Leads
NOK 0
False Positive
90.93%
56,927 Leads
NOK -10,965,594
False Negative
0.01%
6 Leads
NOK 0
True Positive
4.79%
2,996 Leads
NOK 99,107,680
Actual Values
False
True
False
True
Predicted Values
50000
40000
30000
20000
10000