CHURN PREDICTION

Importing Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import warnings
warnings.filterwarnings('ignore')
```

Loading Datasets

In [2]:
```
client_data = pd.read_csv("./Data/client_data.csv")
client_data.head()
```

Out[2]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13

5 rows × 26 columns

In [3]:
```
price_data = pd.read_csv("./Data/price_data.csv")
price_data.head()
```

Out[3]:

	id	price_date	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.266931	
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.266931	
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.266931	
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.266931	
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.266931	

Data Description

In [4]:
```
client_data.shape
```

Out[4]:
```
(14606, 26)
```

In [5]:
```
client_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   id                              14606 non-null  object 
 1   channel_sales                   14606 non-null  object 
 2   cons_12m                        14606 non-null  int64  
 3   cons_gas_12m                    14606 non-null  int64  
 4   cons_last_month                 14606 non-null  int64  
 5   date_activ                      14606 non-null  object 
 6   date_end                        14606 non-null  object 
 7   date_modif_prod                 14606 non-null  object 
 8   date_renewal                    14606 non-null  object 
 9   forecast_cons_12m               14606 non-null  float64
 10  forecast_cons_year              14606 non-null  int64  
 11  forecast_discount_energy        14606 non-null  float64
 12  forecast_meter_rent_12m         14606 non-null  float64
 13  forecast_price_energy_off_peak  14606 non-null  float64
 14  forecast_price_energy_peak      14606 non-null  float64
 15  forecast_price_pow_off_peak     14606 non-null  float64
 16  has_gas                         14606 non-null  object 
 17  imp_cons                        14606 non-null  float64
 18  margin_gross_pow_ele            14606 non-null  float64
 19  margin_net_pow_ele              14606 non-null  float64
 20  nb_prod_act                     14606 non-null  int64  
 21  net_margin                      14606 non-null  float64
 22  num_years_antig                 14606 non-null  int64  
 23  origin_up                       14606 non-null  object 
 24  pow_max                         14606 non-null  float64
 25  churn                           14606 non-null  int64  
dtypes: float64(11), int64(7), object(8)
memory usage: 2.9+ MB
```

In [6]:
```
price_data.shape
```

Out[6]:
```
(193002, 8)
```

In [7]:
```
price_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 193002 entries, 0 to 193001
Data columns (total 8 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   id                  193002 non-null  object 
 1   price_date          193002 non-null  object 
 2   price_off_peak_var  193002 non-null  float64
 3   price_peak_var      193002 non-null  float64
 4   price_mid_peak_var  193002 non-null  float64
 5   price_off_peak_fix  193002 non-null  float64
 6   price_peak_fix      193002 non-null  float64
 7   price_mid_peak_fix  193002 non-null  float64
dtypes: float64(6), object(2)
memory usage: 11.8+ MB
```

Data Statistics

In [8]:
```
client_data.describe()
```

Out[8]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	forecast_discount_energy	forecast_
count	1.460600e+04	1.460600e+04	14606.000000	14606.000000	14606.000000	14606.000000	
mean	1.592203e+05	2.809238e+04	16090.269752	1868.614880	1399.762906	0.966726	
std	5.734653e+05	1.629731e+05	64364.196422	2387.571531	3247.786255	5.108289	
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	
25%	5.674750e+03	0.000000e+00	0.000000	494.995000	0.000000	0.000000	
50%	1.411550e+04	0.000000e+00	792.500000	1112.875000	314.000000	0.000000	
75%	4.076375e+04	0.000000e+00	3383.000000	2401.790000	1745.750000	0.000000	
max	6.207104e+06	4.154590e+06	771203.000000	82902.830000	175375.000000	30.000000	

In [9]:
```
price_data.describe()
```

Out[9]:

	price_off_peak_var	price_peak_var	price_mid_peak_var	price_off_peak_fix	price_peak_fix	price_mid_peak_fix
count	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000	193002.000000
mean	0.141027	0.054630	0.030496	43.334477	10.622875	6.409984
std	0.025032	0.049924	0.036298	5.410297	12.841895	7.773592
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.125976	0.000000	0.000000	40.728885	0.000000	0.000000
50%	0.146033	0.085483	0.000000	44.266930	0.000000	0.000000
75%	0.151635	0.101673	0.072558	44.444710	24.339581	16.226389
max	0.280700	0.229788	0.114102	59.444710	36.490692	17.458221

Checking Null Values

In [10]:
```
client_data.isna().sum()
```

Out[10]:
```
id                                0
channel_sales                     0
cons_12m                          0
cons_gas_12m                      0
cons_last_month                   0
date_activ                        0
date_end                          0
date_modif_prod                   0
date_renewal                      0
forecast_cons_12m                 0
forecast_cons_year                0
forecast_discount_energy          0
forecast_meter_rent_12m           0
forecast_price_energy_off_peak    0
forecast_price_energy_peak        0
forecast_price_pow_off_peak       0
has_gas                           0
imp_cons                          0
margin_gross_pow_ele              0
margin_net_pow_ele                0
nb_prod_act                       0
net_margin                        0
num_years_antig                   0
origin_up                         0
pow_max                           0
churn                             0
dtype: int64
```

In [11]:
```
price_data.isna().sum()
```

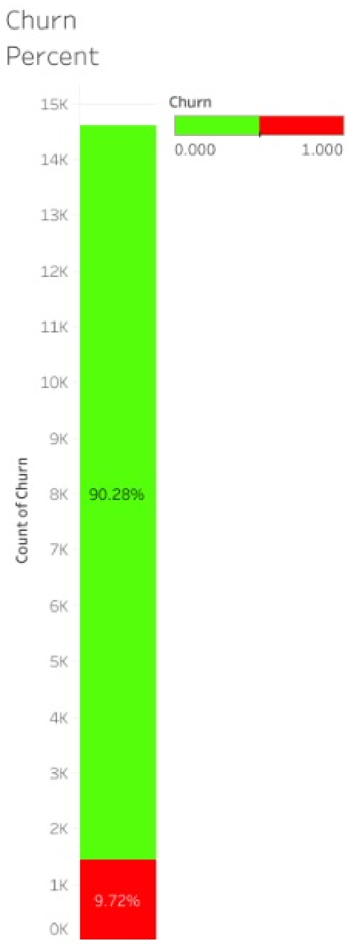
Out[11]:
```
id                    0
price_date            0
price_off_peak_var    0
price_peak_var        0
price_mid_peak_var    0
price_off_peak_fix    0
price_peak_fix        0
price_mid_peak_fix    0
dtype: int64
```

EDA

Conducted a thorough investigation of the dataset using Tableau. The key visualizations and insights from this Tableau-based EDA are embedded below to provide context and support our subsequent modeling efforts.

Churn Percent

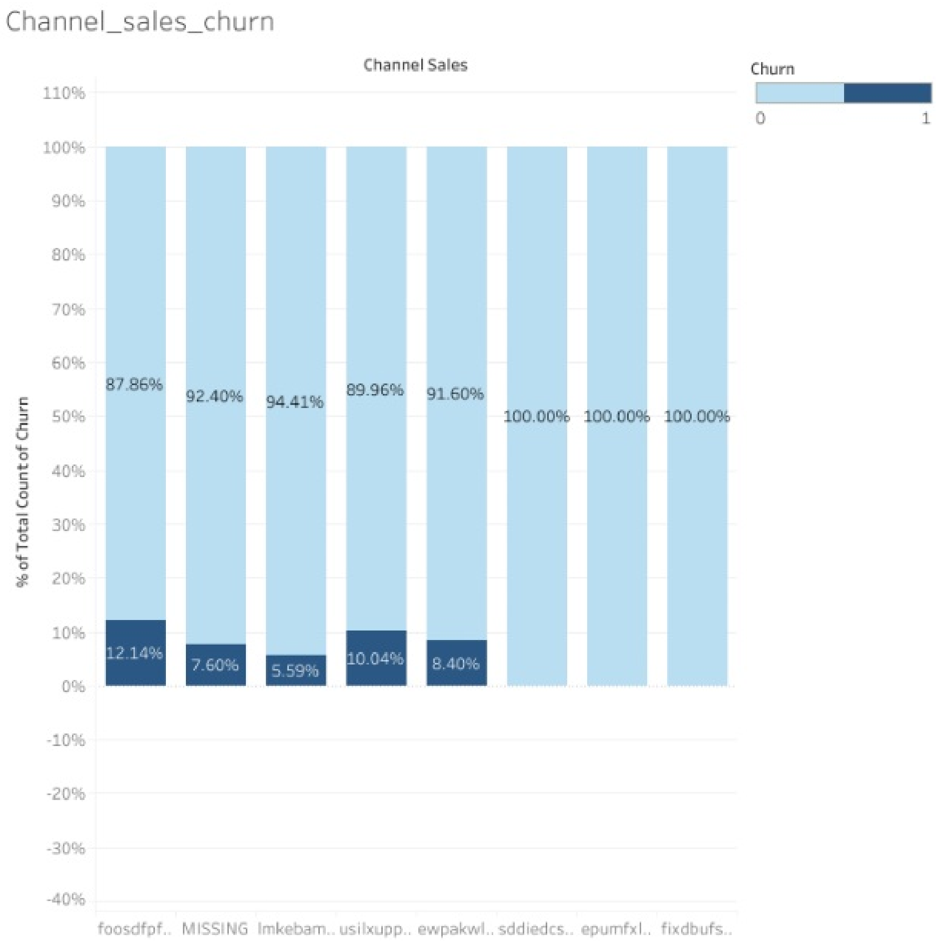
In [12]:
```
img = mpimg.imread("./images/Churn Percent.png")
plt.figure(figsize=(18,8))
plt.imshow(img)
plt.axis('off')
plt.show()
```



From the above chart we can see that approximately 10% of the customers are switching

Channel Wise Churn Percent

In [13]:
```
img = mpimg.imread("./images/Channel_sales_churn.png")
plt.figure(figsize=(18,8))
plt.imshow(img)
plt.axis('off')
plt.show()
```

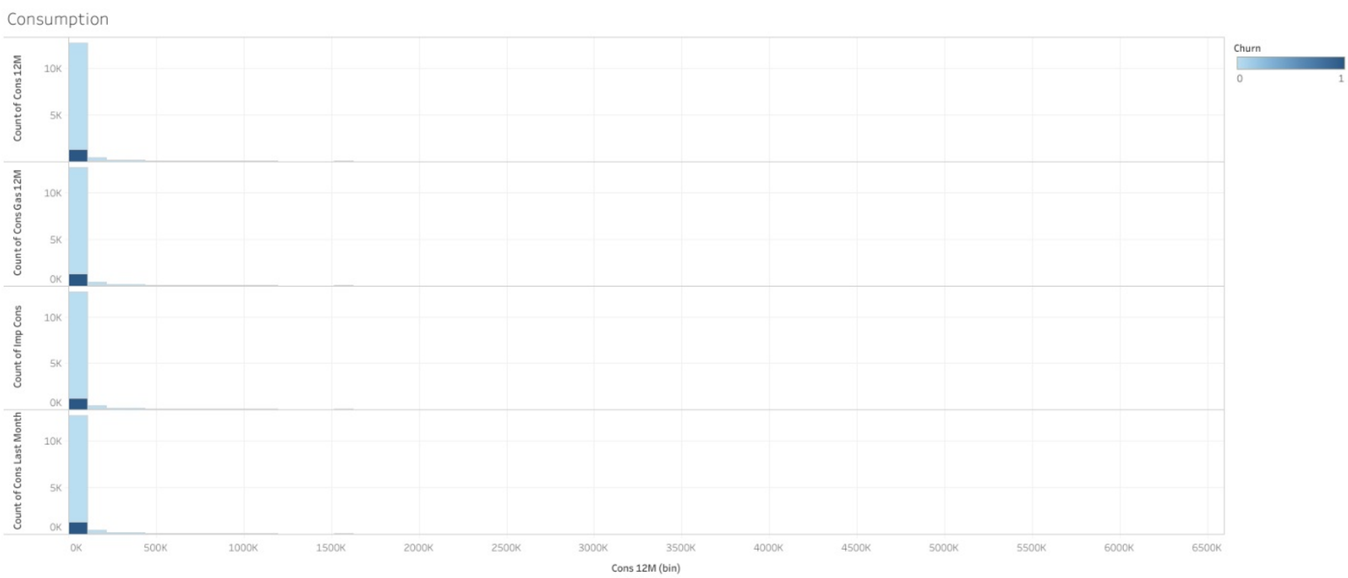


From the above chart we can see that maximum customers from the "foosdfpfkusacimwkcsosbicdxkicaua" and "usilxuppasemubllopkaafesmlibmsdf" channel

Consumption based Churn Percent

In [14]:

```python
img = mpimg.imread("./images/Consumption.png")
plt.figure(figsize=(18,12))
plt.imshow(img)
plt.axis('off')
plt.show()
```



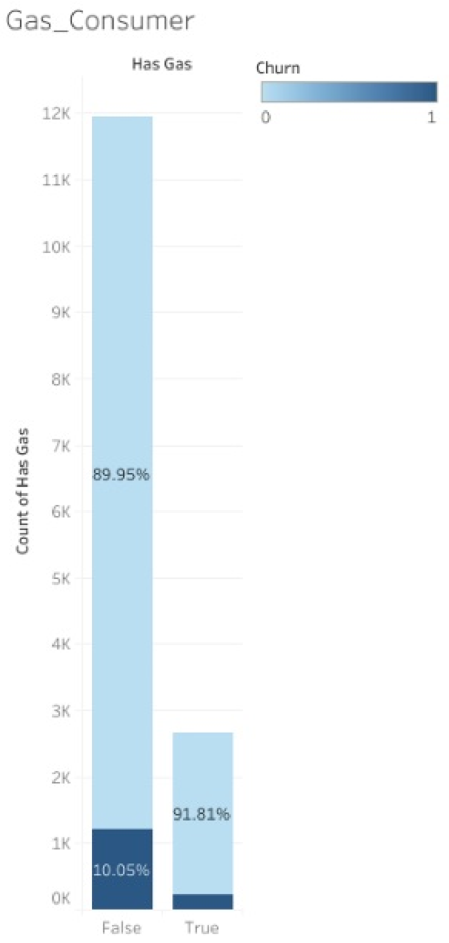
From the above chart we can see that customers having consumption between 0 and 100k are switching.

Gas Consumer Churn Percent

In [15]:

```python
img = mpimg.imread("./images/Gas_Consumer.png")
plt.figure(figsize=(10,8))
plt.imshow(img)
plt.axis('off')
```

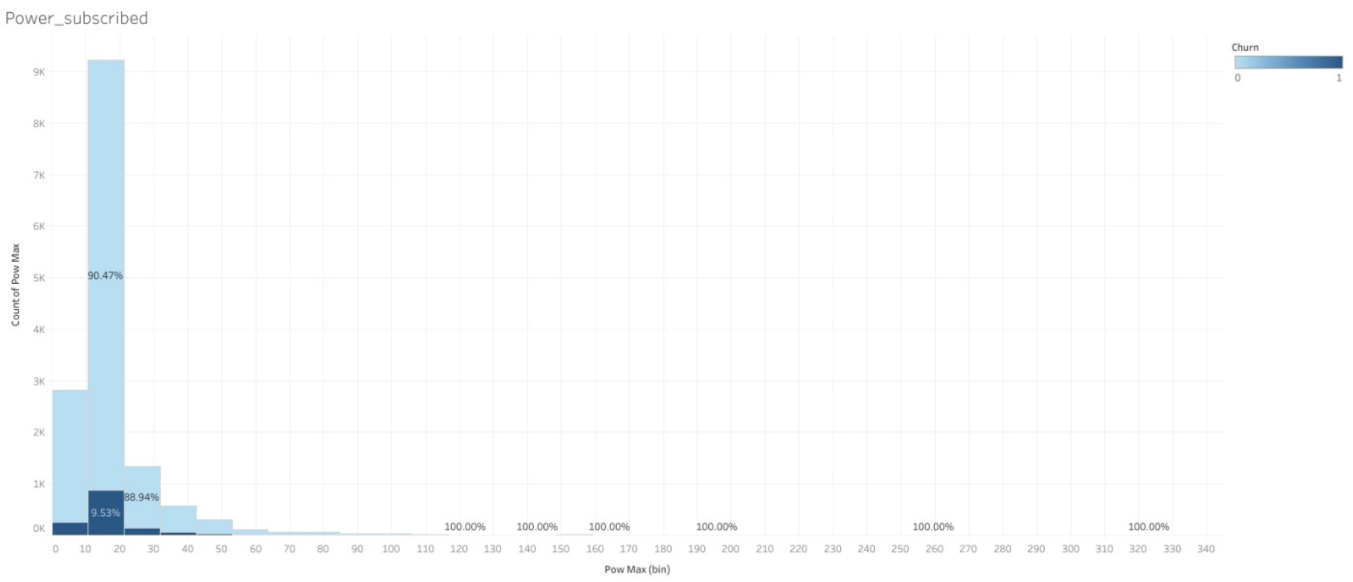
```python
plt.show()
```



From this chart we can see that customers who take electricity but does not take gas suppy has the higher percent of switching.

Subscribed Power Churn Percent

In [16]:
```python
img = mpimg.imread("./images/Power_subscribed.png")
plt.figure(figsize=(18,8))
plt.imshow(img)
plt.axis('off')
plt.show()
```



From this chart we can see that customers with subscribed power between 10 to 20 are switching.

Feature Engineering

In [17]:
```python
# Converting date columns to datetime format
date_cols = ['date_activ', 'date_end', 'date_renewal', 'date_modif_prod']
for col in date_cols:
```

```
    client_data[col] = pd.to_datetime(client_data[col])

price_data["price_date"] = pd.to_datetime(price_data["price_date"])
```

Difference between off-peak prices in December and preceding January

In [18]:
```
# Group off-peak prices by companies and month
monthly_price_by_id = price_data.groupby(['id', 'price_date']).agg({'price_off_peak_var': 'mean', 'price_off_pea

# Get january and december prices
jan_prices = monthly_price_by_id.groupby('id').first().reset_index()
dec_prices = monthly_price_by_id.groupby('id').last().reset_index()

# Calculate the difference
diff = pd.merge(dec_prices.rename(columns={'price_off_peak_var': 'dec_1', 'price_off_peak_fix': 'dec_2'}), jan_
diff['offpeak_diff_dec_january_energy'] = diff['dec_1'] - diff['price_off_peak_var']
diff['offpeak_diff_dec_january_power'] = diff['dec_2'] - diff['price_off_peak_fix']
diff = diff[['id', 'offpeak_diff_dec_january_energy','offpeak_diff_dec_january_power']]
diff.head()
```

Out[18]:

	id	offpeak_diff_dec_january_energy	offpeak_diff_dec_january_power
0	0002203ffbb812588b632b9e628cc38d	-0.006192	0.162916
1	0004351ebdd665e6ee664792efc4fd13	-0.004104	0.177779
2	0010bcc39e42b3c2131ed2ce55246e3c	0.050443	1.500000
3	0010ee3855fdea87602a5b7aba8e42de	-0.010018	0.162916
4	00114d74e963e47177db89bc70108537	-0.003994	-0.000001

In [19]:
```
df = pd.merge(client_data, diff, on='id')
df.head()
```

Out[19]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13

5 rows × 28 columns

Average price changes across periods

In [20]:
```
# Aggregate average prices per period by company
mean_prices = price_data.groupby(['id']).agg({
    'price_off_peak_var': 'mean',
    'price_peak_var': 'mean',
    'price_mid_peak_var': 'mean',
    'price_off_peak_fix': 'mean',
    'price_peak_fix': 'mean',
    'price_mid_peak_fix': 'mean'
}).reset_index()
```

In [21]:
```
# Calculate the mean difference between consecutive periods
mean_prices['off_peak_peak_var_mean_diff'] = mean_prices['price_off_peak_var'] - mean_prices['price_peak_var']
mean_prices['peak_mid_peak_var_mean_diff'] = mean_prices['price_peak_var'] - mean_prices['price_mid_peak_var']
mean_prices['off_peak_mid_peak_var_mean_diff'] = mean_prices['price_off_peak_var'] - mean_prices['price_mid_peal
mean_prices['off_peak_peak_fix_mean_diff'] = mean_prices['price_off_peak_fix'] - mean_prices['price_peak_fix']
mean_prices['peak_mid_peak_fix_mean_diff'] = mean_prices['price_peak_fix'] - mean_prices['price_mid_peak_fix']
mean_prices['off_peak_mid_peak_fix_mean_diff'] = mean_prices['price_off_peak_fix'] - mean_prices['price_mid_peal
```

In [22]:
```
columns = ['id', 'off_peak_peak_var_mean_diff','peak_mid_peak_var_mean_diff', 'off_peak_mid_peak_var_mean_diff'
df = pd.merge(df, mean_prices[columns], on='id')
df.head()
```

Out[22]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13

5 rows × 34 columns

Max price changes across periods and months

In [23]:
```
# Aggregate average prices per period by company
mean_prices_by_month = price_data.groupby(['id', 'price_date']).agg({'price_off_peak_var': 'mean', 'price_peak_v
```

In [24]:
```
# Calculate the mean difference between consecutive periods
mean_prices_by_month['off_peak_peak_var_mean_diff'] = mean_prices_by_month['price_off_peak_var'] - mean_prices_l
mean_prices_by_month['peak_mid_peak_var_mean_diff'] = mean_prices_by_month['price_peak_var'] - mean_prices_by_m
mean_prices_by_month['off_peak_mid_peak_var_mean_diff'] = mean_prices_by_month['price_off_peak_var'] - mean_pri
mean_prices_by_month['off_peak_peak_fix_mean_diff'] = mean_prices_by_month['price_off_peak_fix'] - mean_prices_l
mean_prices_by_month['peak_mid_peak_fix_mean_diff'] = mean_prices_by_month['price_peak_fix'] - mean_prices_by_m
mean_prices_by_month['off_peak_mid_peak_fix_mean_diff'] = mean_prices_by_month['price_off_peak_fix'] - mean_pri
```

In [25]:
```
# Calculate the maximum monthly difference across time periods
max_diff_across_periods_months = mean_prices_by_month.groupby(['id']).agg({'off_peak_peak_var_mean_diff': 'max'
```

In [26]:
```
columns = ['id','off_peak_peak_var_max_monthly_diff','peak_mid_peak_var_max_monthly_diff','off_peak_mid_peak_va
df = pd.merge(df, max_diff_across_periods_months[columns], on='id')
df.head()
```

Out[26]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	date_activ
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	2013-06-15
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	2009-08-21
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	2010-04-16
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	2010-03-30
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	2010-01-13

5 rows × 40 columns

Tenure

In [27]:
```
df['tenure'] = df.date_end.dt.year - df.date_activ.dt.year
```

In [28]:
```
df.groupby(['tenure']).agg({'churn': 'mean'}).sort_values(by='churn', ascending=False)
```

Out[28]:

tenure	churn
3	0.144612
4	0.126383
5	0.099897
13	0.093023
12	0.085106
6	0.075593
7	0.075025
8	0.058065
11	0.052356
9	0.037037
10	0.027778
2	0.000000

Transforming dates into months

- months_activ = Number of months active until reference date (Jan 2016)
- months_to_end = Number of months of the contract left until reference date (Jan 2016)
- months_modif_prod = Number of months since last modification until reference date (Jan 2016)
- months_renewal = Number of months since last renewal until reference date (Jan 2016)

In [29]:
```
def convert_months(reference_date, df, column):
    months = (reference_date.year - df[column].dt.year)*12 + (reference_date.month - df[column].dt.month)
    return months
```

In [30]:
```
# Create reference date
reference_date = pd.to_datetime('2016-01-01')

# Create columns
df['months_activ'] = convert_months(reference_date, df, 'date_activ')
df['months_to_end'] = -convert_months(reference_date, df, 'date_end')
df['months_modif_prod'] = convert_months(reference_date, df, 'date_modif_prod')
df['months_renewal'] = convert_months(reference_date, df, 'date_renewal')
```

In [31]:
```
remove = ['date_activ','date_end','date_modif_prod','date_renewal']
df = df.drop(columns=remove)
df.head()
```

Out[31]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	forecast_co
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	

5 rows × 41 columns

In [32]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 41 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   id                                  14606 non-null  object 
 1   channel_sales                       14606 non-null  object 
 2   cons_12m                            14606 non-null  int64  
 3   cons_gas_12m                        14606 non-null  int64  
 4   cons_last_month                     14606 non-null  int64  
 5   forecast_cons_12m                   14606 non-null  float64
 6   forecast_cons_year                  14606 non-null  int64  
 7   forecast_discount_energy            14606 non-null  float64
 8   forecast_meter_rent_12m             14606 non-null  float64
 9   forecast_price_energy_off_peak      14606 non-null  float64
 10  forecast_price_energy_peak          14606 non-null  float64
 11  forecast_price_pow_off_peak         14606 non-null  float64
 12  has_gas                             14606 non-null  object 
 13  imp_cons                            14606 non-null  float64
 14  margin_gross_pow_ele                14606 non-null  float64
 15  margin_net_pow_ele                  14606 non-null  float64
 16  nb_prod_act                         14606 non-null  int64  
 17  net_margin                          14606 non-null  float64
 18  num_years_antig                     14606 non-null  int64  
 19  origin_up                           14606 non-null  object 
 20  pow_max                             14606 non-null  float64
 21  churn                               14606 non-null  int64  
 22  offpeak_diff_dec_january_energy     14606 non-null  float64
 23  offpeak_diff_dec_january_power      14606 non-null  float64
 24  off_peak_peak_var_mean_diff         14606 non-null  float64
 25  peak_mid_peak_var_mean_diff         14606 non-null  float64
 26  off_peak_mid_peak_var_mean_diff     14606 non-null  float64
 27  off_peak_peak_fix_mean_diff         14606 non-null  float64
 28  peak_mid_peak_fix_mean_diff         14606 non-null  float64
 29  off_peak_mid_peak_fix_mean_diff     14606 non-null  float64
 30  off_peak_peak_var_max_monthly_diff  14606 non-null  float64
 31  peak_mid_peak_var_max_monthly_diff  14606 non-null  float64
 32  off_peak_mid_peak_var_max_monthly_diff  14606 non-null  float64
 33  off_peak_peak_fix_max_monthly_diff  14606 non-null  float64
 34  peak_mid_peak_fix_max_monthly_diff  14606 non-null  float64
 35  off_peak_mid_peak_fix_max_monthly_diff  14606 non-null  float64
 36  tenure                              14606 non-null  int32  
 37  months_activ                        14606 non-null  int32  
 38  months_to_end                       14606 non-null  int32  
 39  months_modif_prod                   14606 non-null  int32  
 40  months_renewal                      14606 non-null  int32  
dtypes: float64(25), int32(5), int64(7), object(4)
memory usage: 4.3+ MB
```

In [33]:
```
df = pd.get_dummies(df, columns=["origin_up"], prefix="origin_up", dtype = int)
df
```

Out[33]:

	id	channel_sales	cons_12m	cons_gas_12m	cons_last_month	forecas
0	24011ae4ebbe3035111d65fa7c15bc57	foosdfpfkusacimwkcsosbicdxkicaua	0	54946	0	
1	d29c2c54acc38ff3c0614d0a653813dd	MISSING	4660	0	0	
2	764c75f661154dac3a6c254cd082ea7d	foosdfpfkusacimwkcsosbicdxkicaua	544	0	0	
3	bba03439a292a1e166f80264c16191cb	lmkebamcaaclubfxadlmueccxoimlema	1584	0	0	
4	149d57cf92fc41cf94415803a877cb4b	MISSING	4425	0	526	
...	...	...	...	...	...	
14601	18463073fb097fc0ac5d3e040f356987	foosdfpfkusacimwkcsosbicdxkicaua	32270	47940	0	
14602	d0a6f71671571ed83b2645d23af6de00	foosdfpfkusacimwkcsosbicdxkicaua	7223	0	181	
14603	10e6828ddd62cbcf687cb74928c4c2d2	foosdfpfkusacimwkcsosbicdxkicaua	1844	0	179	
14604	1cf20fd6206d7678d5bcafd28c53b4db	foosdfpfkusacimwkcsosbicdxkicaua	131	0	0	
14605	563dde550fd624d7352f3de77c0cdfcd	MISSING	8730	0	0	

14606 rows × 46 columns

In [34]:
```
df = pd.get_dummies(df, columns=["channel_sales"], prefix="channel_sales", dtype = int)
df
```

Out[34]:

	id	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	fc
0	24011ae4ebbe3035111d65fa7c15bc57	0	54946	0	0.00	0	
1	d29c2c54acc38ff3c0614d0a653813dd	4660	0	0	189.95	0	
2	764c75f661154dac3a6c254cd082ea7d	544	0	0	47.96	0	
3	bba03439a292a1e166f80264c16191cb	1584	0	0	240.04	0	
4	149d57cf92fc41cf94415803a877cb4b	4425	0	526	445.75	526	
...	...	...	...	...	...	...	
14601	18463073fb097fc0ac5d3e040f356987	32270	47940	0	4648.01	0	
14602	d0a6f71671571ed83b2645d23af6de00	7223	0	181	631.69	181	
14603	10e6828ddd62cbcf687cb74928c4c2d2	1844	0	179	190.39	179	
14604	1cf20fd6206d7678d5bcafd28c53b4db	131	0	0	19.34	0	
14605	563dde550fd624d7352f3de77c0cdfcd	8730	0	0	762.41	0	

14606 rows × 53 columns

In [35]:

```
df.has_gas.replace({'f': 0, 't': 1}, inplace=True)
df
```

Out[35]:

	id	cons_12m	cons_gas_12m	cons_last_month	forecast_cons_12m	forecast_cons_year	fc
0	24011ae4ebbe3035111d65fa7c15bc57	0	54946	0	0.00	0	
1	d29c2c54acc38ff3c0614d0a653813dd	4660	0	0	189.95	0	
2	764c75f661154dac3a6c254cd082ea7d	544	0	0	47.96	0	
3	bba03439a292a1e166f80264c16191cb	1584	0	0	240.04	0	
4	149d57cf92fc41cf94415803a877cb4b	4425	0	526	445.75	526	
...	...	...	...	...	...	...	
14601	18463073fb097fc0ac5d3e040f356987	32270	47940	0	4648.01	0	
14602	d0a6f71671571ed83b2645d23af6de00	7223	0	181	631.69	181	
14603	10e6828ddd62cbcf687cb74928c4c2d2	1844	0	179	190.39	179	
14604	1cf20fd6206d7678d5bcafd28c53b4db	131	0	0	19.34	0	
14605	563dde550fd624d7352f3de77c0cdfcd	8730	0	0	762.41	0	

14606 rows × 53 columns

In [36]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14606 entries, 0 to 14605
Data columns (total 53 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   id                                            14606 non-null  object 
 1   cons_12m                                      14606 non-null  int64  
 2   cons_gas_12m                                  14606 non-null  int64  
 3   cons_last_month                               14606 non-null  int64  
 4   forecast_cons_12m                             14606 non-null  float64
 5   forecast_cons_year                            14606 non-null  int64  
 6   forecast_discount_energy                      14606 non-null  float64
 7   forecast_meter_rent_12m                       14606 non-null  float64
 8   forecast_price_energy_off_peak                14606 non-null  float64
 9   forecast_price_energy_peak                    14606 non-null  float64
 10  forecast_price_pow_off_peak                   14606 non-null  float64
 11  has_gas                                       14606 non-null  int64  
 12  imp_cons                                      14606 non-null  float64
 13  margin_gross_pow_ele                          14606 non-null  float64
 14  margin_net_pow_ele                            14606 non-null  float64
 15  nb_prod_act                                   14606 non-null  int64  
 16  net_margin                                    14606 non-null  float64
 17  num_years_antig                               14606 non-null  int64  
 18  pow_max                                       14606 non-null  float64
 19  churn                                         14606 non-null  int64  
 20  offpeak_diff_dec_january_energy               14606 non-null  float64
 21  offpeak_diff_dec_january_power                14606 non-null  float64
 22  off_peak_peak_var_mean_diff                   14606 non-null  float64
 23  peak_mid_peak_var_mean_diff                   14606 non-null  float64
 24  off_peak_mid_peak_var_mean_diff               14606 non-null  float64
 25  off_peak_peak_fix_mean_diff                   14606 non-null  float64
 26  peak_mid_peak_fix_mean_diff                   14606 non-null  float64
 27  off_peak_mid_peak_fix_mean_diff               14606 non-null  float64
 28  off_peak_peak_var_max_monthly_diff            14606 non-null  float64
 29  peak_mid_peak_var_max_monthly_diff            14606 non-null  float64
 30  off_peak_mid_peak_var_max_monthly_diff        14606 non-null  float64
 31  off_peak_peak_fix_max_monthly_diff            14606 non-null  float64
 32  peak_mid_peak_fix_max_monthly_diff            14606 non-null  float64
 33  off_peak_mid_peak_fix_max_monthly_diff        14606 non-null  float64
 34  tenure                                        14606 non-null  int32  
 35  months_activ                                  14606 non-null  int32  
 36  months_to_end                                 14606 non-null  int32  
 37  months_modif_prod                             14606 non-null  int32  
 38  months_renewal                                14606 non-null  int32  
 39  origin_up_MISSING                             14606 non-null  int32  
 40  origin_up_ewxeelcelemmiwuafmddpobolfuxioce    14606 non-null  int32  
 41  origin_up_kamkkxfxxuwbdslkwifmmcsiusiuosws    14606 non-null  int32  
 42  origin_up_ldkssxwpmemidmecebumciepifcamkci    14606 non-null  int32  
 43  origin_up_lxidpiddsbxsbosboudacockeimpuepw    14606 non-null  int32  
 44  origin_up_usapbepcfoloekilkwsdiboslwaxobdp    14606 non-null  int32  
 45  channel_sales_MISSING                         14606 non-null  int32  
 46  channel_sales_epumfxlbckeskwekxbiuasklxalciiuu  14606 non-null  int32  
 47  channel_sales_ewpakwlliwisiwduibdlfmalxowmwpci  14606 non-null  int32  
 48  channel_sales_fixdbufsefwooaasfcxdxadsiekoceaa  14606 non-null  int32  
 49  channel_sales_foosdfpfkusacimwkcsosbicdxkicaua  14606 non-null  int32  
 50  channel_sales_lmkebamcaaclubfxadlmueccxoimlema  14606 non-null  int32  
 51  channel_sales_sddiedcslfslkckwlfkdpoeeailfpeds  14606 non-null  int32  
 52  channel_sales_usilxuppasemubllopkaafesmlibmsdf  14606 non-null  int32  
dtypes: float64(25), int32(19), int64(8), object(1)
memory usage: 4.8+ MB
```

Modeling the Data

In [37]:
```
X = df.drop(['id','churn'], axis=1)
X.shape
```

Out[37]: (14606, 51)

In [38]:
```
y = df.churn
y.shape
```

Out[38]: (14606,)

Splitting dataset into Training and Testing

In [39]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

In [40]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(11684, 51)
(2922, 51)
(11684,)
(2922,)
```

Scaling the Data

In [41]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Building the ANN Model

In [42]:
```
model = Sequential([
    Flatten(),
    Dense(45, activation='relu'),
    Dropout(0.2),
    Dense(35, activation='relu'),
    Dense(25, activation='relu'),
    Dense(10, activation='relu'),
    Dense(1, activation='sigmoid'),
])
```

In [43]:
```
model.compile(
    optimizer = 'adam',
    loss = 'binary_crossentropy',
    metrics = ['accuracy']
)
```

In [44]:
```
history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=20, batch_size=32)
```

```
Epoch 1/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - accuracy: 0.8793 - loss: 0.3973 - val_accuracy: 0.9028 - val_loss: 0.
3105
Epoch 2/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9050 - loss: 0.3088 - val_accuracy: 0.9028 - val_loss: 0.
3128
Epoch 3/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9000 - loss: 0.3177 - val_accuracy: 0.9028 - val_loss: 0.
3093
Epoch 4/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9082 - loss: 0.2989 - val_accuracy: 0.9028 - val_loss: 0.
3085
Epoch 5/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9016 - loss: 0.3137 - val_accuracy: 0.9028 - val_loss: 0.
3086
Epoch 6/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9042 - loss: 0.3033 - val_accuracy: 0.9028 - val_loss: 0.
3073
Epoch 7/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9068 - loss: 0.2968 - val_accuracy: 0.9028 - val_loss: 0.
3147
Epoch 8/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9061 - loss: 0.3008 - val_accuracy: 0.9028 - val_loss: 0.
3065
Epoch 9/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9064 - loss: 0.2977 - val_accuracy: 0.9028 - val_loss: 0.
3069
Epoch 10/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9036 - loss: 0.3018 - val_accuracy: 0.9028 - val_loss: 0.
3070
Epoch 11/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9088 - loss: 0.2905 - val_accuracy: 0.9028 - val_loss: 0.
3099
Epoch 12/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9052 - loss: 0.2998 - val_accuracy: 0.9028 - val_loss: 0.
3074
Epoch 13/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9066 - loss: 0.2944 - val_accuracy: 0.9028 - val_loss: 0.
3088
Epoch 14/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9008 - loss: 0.3079 - val_accuracy: 0.9028 - val_loss: 0.
3088
Epoch 15/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9040 - loss: 0.2986 - val_accuracy: 0.9028 - val_loss: 0.
3046
Epoch 16/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9014 - loss: 0.3051 - val_accuracy: 0.9028 - val_loss: 0.
3072
Epoch 17/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9064 - loss: 0.2894 - val_accuracy: 0.9028 - val_loss: 0.
3074
Epoch 18/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8975 - loss: 0.3100 - val_accuracy: 0.9028 - val_loss: 0.
3069
Epoch 19/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8977 - loss: 0.3110 - val_accuracy: 0.9028 - val_loss: 0.
3078
Epoch 20/20
366/366 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9064 - loss: 0.2891 - val_accuracy: 0.9028 - val_loss: 0.
3070
```

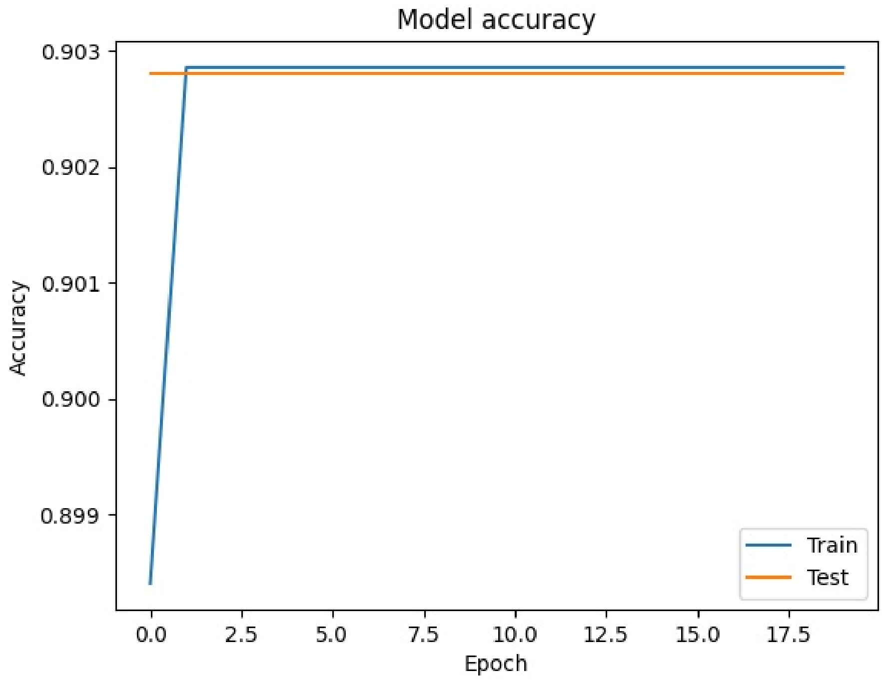
In [45]:
```python
y_pred_values = model.predict(X_test)
```

```
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

In [46]:
```python
y_pred = []
for value in y_pred_values:
    if value > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

Model Evaluation

In [47]:
```python
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='best')
plt.show()
```



In [48]:
```
model.evaluate(X_test, y_test)
```

```
92/92 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9005 - loss: 0.3118
```

Out[48]:
```
[0.30697688460350037, 0.902806282043457]
```

In [49]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95      2638
           1       0.00      0.00      0.00       284

    accuracy                           0.90      2922
   macro avg       0.45      0.50      0.47      2922
weighted avg       0.82      0.90      0.86      2922
```

In [50]:
```
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True)
plt.title(f"Accuracy for ANN: {round(accuracy_score(y_test, y_pred),2)}")
plt.xlabel("Predicted")
plt.ylabel("Truth")
plt.show()
```