Importing modules

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

Importing data

In [2]:
```
ml_tr=pd.read_csv("ml_case_training_data.csv")
ml_tr_hist=pd.read_csv("ml_case_training_hist_data.csv")
ml_tr_out=pd.read_csv("ml_case_training_output.csv")
```

Examining transaction data

Exploratory data analysis

In [3]:
```
ml_tr.head()
```

Out[3]:

	id	activity_new	campaign_disc_ele	
0	48ada52261e7cf58715202705a0451c9	esoiiifxdlbkcsluxmfuacbdckommixw	NaN	lmkebamc
1	24011ae4ebbe3035111d65fa7c15bc57	NaN	NaN	foosdfp
2	d29c2c54acc38ff3c0614d0a653813dd	NaN	NaN	
3	764c75f661154dac3a6c254cd082ea7d	NaN	NaN	foosdfp
4	bba03439a292a1e166f80264c16191cb	NaN	NaN	lmkebamc

5 rows × 32 columns

In [4]:
```
ml_tr_hist.head()
```

Out[4]:

	id	price_date	price_p1_var	price_p2_var	price_p3_var	price_p1_
0	038af19179925da21a25619c5a24b745	2015-01-01	0.151367	0.0	0.0	44.2669
1	038af19179925da21a25619c5a24b745	2015-02-01	0.151367	0.0	0.0	44.2669
2	038af19179925da21a25619c5a24b745	2015-03-01	0.151367	0.0	0.0	44.2669
3	038af19179925da21a25619c5a24b745	2015-04-01	0.149626	0.0	0.0	44.2669
4	038af19179925da21a25619c5a24b745	2015-05-01	0.149626	0.0	0.0	44.2669

In [5]:
```
ml_tr_out.head()
```

Out[5]:

	id	churn
0	48ada52261e7cf58715202705a0451c9	0
1	24011ae4ebbe3035111d65fa7c15bc57	1
2	d29c2c54acc38ff3c0614d0a653813dd	0
3	764c75f661154dac3a6c254cd082ea7d	0
4	bba03439a292a1e166f80264c16191cb	0

Many inconsistancies. Needs work.

In [6]:
```
ml_tr.loc[ml_tr.has_gas=='t',"has_gas"]=1
ml_tr.loc[ml_tr.has_gas=='f',"has_gas"]=0
```

In [7]:
```
ml_tr = ml_tr.fillna(ml_tr.mean())
ml_tr=ml_tr.drop(columns=["campaign_disc_ele"])
```

dropping dulpicates

In [8]:
```
ml_tr=ml_tr.drop_duplicates('id',keep='first')
```

Converting types

In [9]:
```
ml_tr['date_renewal']= pd.to_datetime(ml_tr['date_renewal'])
ml_tr['date_modif_prod']= pd.to_datetime(ml_tr['date_modif_prod'])
ml_tr['date_first_activ']= pd.to_datetime(ml_tr['date_first_activ'])
ml_tr['date_end']= pd.to_datetime(ml_tr['date_end'])
ml_tr['date_activ']= pd.to_datetime(ml_tr['date_activ'])
```

In [10]:
```
ml_tr_out.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16096 entries, 0 to 16095
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      16096 non-null  object
 1   churn   16096 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 251.6+ KB
```

No data missing.

In [11]:
```
ml_tr_hist.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_p1_var  191643 non-null  float64
```

```
 3   price_p2_var  191643 non-null  float64
 4   price_p3_var  191643 non-null  float64
 5   price_p1_fix  191643 non-null  float64
 6   price_p2_fix  191643 non-null  float64
 7   price_p3_fix  191643 non-null  float64
dtypes: float64(6), object(2)
memory usage: 11.8+ MB
```

Data clearning needed.

In [12]:
```
ml_tr_hist=ml_tr_hist.dropna()
ml_tr_hist.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 191643 entries, 0 to 193001
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   id            191643 non-null  object 
 1   price_date    191643 non-null  object 
 2   price_p1_var  191643 non-null  float64
 3   price_p2_var  191643 non-null  float64
 4   price_p3_var  191643 non-null  float64
 5   price_p1_fix  191643 non-null  float64
 6   price_p2_fix  191643 non-null  float64
 7   price_p3_fix  191643 non-null  float64
dtypes: float64(6), object(2)
memory usage: 13.2+ MB
```

Feature Engineering

In [13]:
```
ml_tr = pd.merge(left=ml_tr, right=ml_tr_out, how='left', left_on='id', right_on='id
```

Sub-task 1: Think through what key drivers of churn could be for our client

correlation mattrix

In [14]:
```
ml_tr.corr()
```

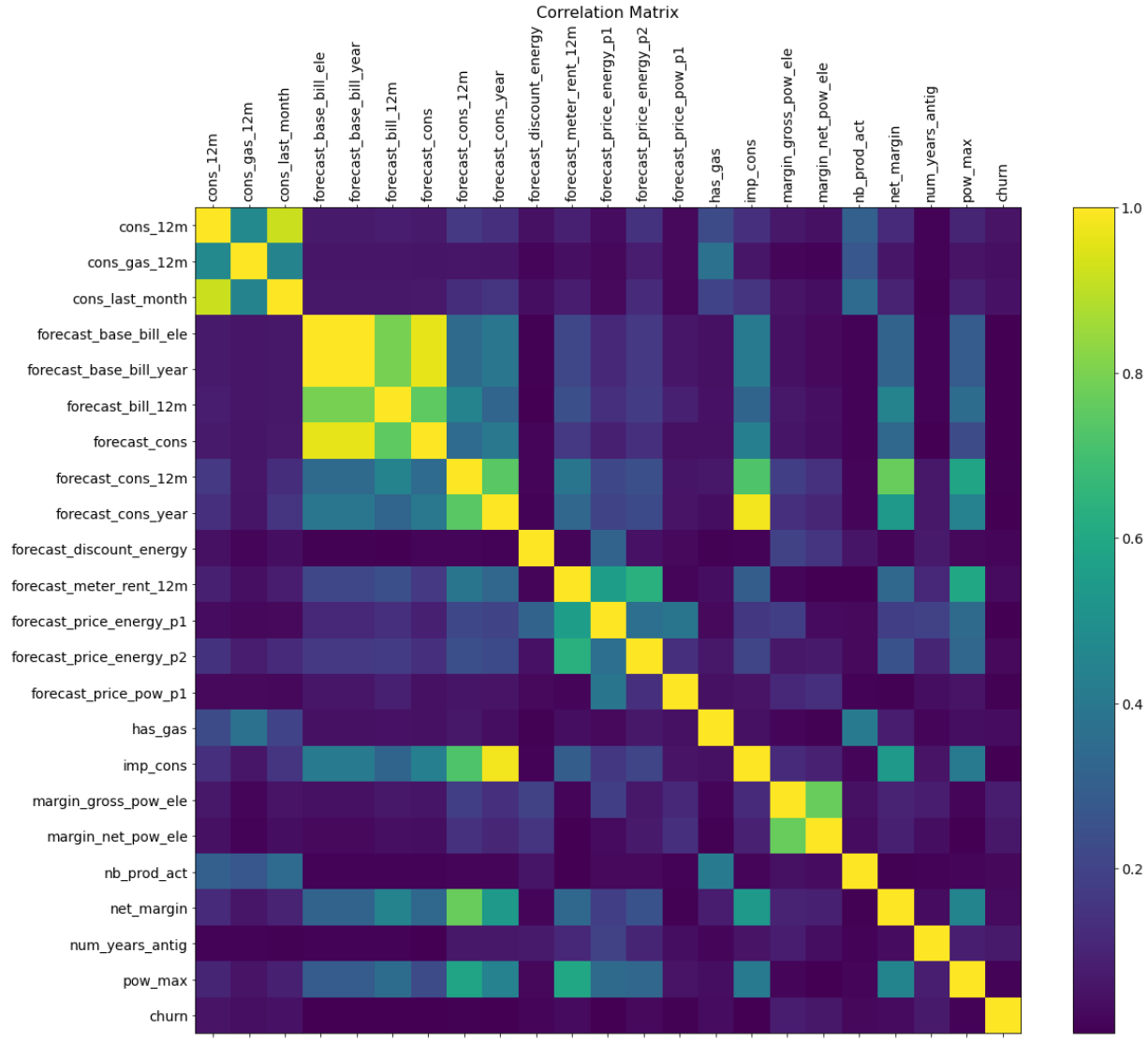
Out[14]:

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_l
cons_12m	1.000000	0.471233	0.919545	0.071443	
cons_gas_12m	0.471233	1.000000	0.447209	0.061064	
cons_last_month	0.919545	0.447209	1.000000	0.066250	
forecast_base_bill_ele	0.071443	0.061064	0.066250	1.000000	
forecast_base_bill_year	0.071443	0.061064	0.066250	1.000000	
forecast_bill_12m	0.080056	0.059547	0.065208	0.794776	
forecast_cons	0.071527	0.054740	0.066546	0.964402	
forecast_cons_12m	0.165168	0.059525	0.129574	0.344620	
forecast_cons_year	0.139526	0.057619	0.151476	0.393361	

	cons_12m	cons_gas_12m	cons_last_month	forecast_base_bill_ele	forecast_l
forecast_discount_energy	-0.043551	-0.014407	-0.037699	0.005792	
forecast_meter_rent_12m	0.085996	0.040327	0.076066	0.214113	
forecast_price_energy_p1	-0.033425	-0.021608	-0.024195	-0.116035	
forecast_price_energy_p2	0.146229	0.075628	0.122922	0.165854	
forecast_price_pow_p1	-0.025326	-0.026212	-0.020017	0.062149	
has_gas	0.229761	0.372771	0.202702	0.046099	
imp_cons	0.139353	0.060609	0.153861	0.414904	
margin_gross_pow_ele	-0.065184	-0.016866	-0.054069	-0.044562	
margin_net_pow_ele	-0.045558	-0.008242	-0.037665	-0.027109	
nb_prod_act	0.308567	0.272005	0.350711	0.010411	
net_margin	0.119910	0.058928	0.096343	0.320016	
num_years_antig	0.008810	-0.008626	0.004860	0.008122	
pow_max	0.102422	0.052365	0.089565	0.291136	
churn	-0.051759	-0.040880	-0.046931	0.000242	

23 rows × 23 columns

In [15]:
```
df=ml_tr.copy()
f = plt.figure(figsize=(19, 15))
plt.matshow(df.corr().abs(), fignum=f.number)
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
plt.yticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16);
```



Selecting imporant features as having corr moire than 0.6

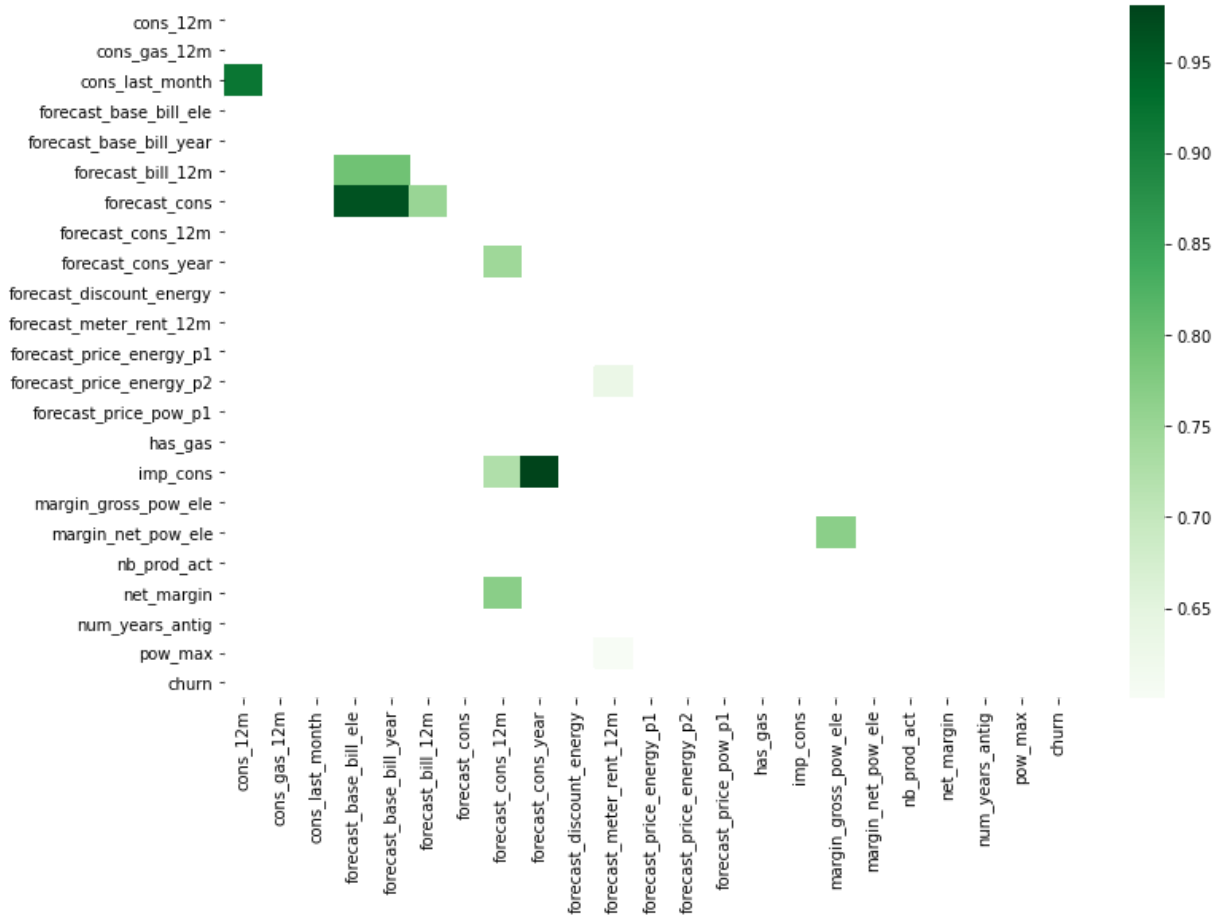
In [16]:
```
corr = df.corr().abs()

corr = corr.where(np.tril(np.ones(corr.shape)).astype(np.bool))

kot = corr[corr>=.6]
kot=kot[kot<1]


plt.figure(figsize=(12,8))
sns.heatmap(kot, cmap="Greens")
```

Out[16]: <AxesSubplot:>



In [17]:
```
data=kot.unstack().sort_values().drop_duplicates()
print(data)
```

```
forecast_meter_rent_12m  pow_max                    0.600566
                         forecast_price_energy_p2   0.632863
forecast_cons_12m        imp_cons                   0.725550
                         forecast_cons_year         0.746076
forecast_bill_12m        forecast_cons              0.751430
margin_gross_pow_ele     margin_net_pow_ele         0.766521
forecast_cons_12m        net_margin                 0.768609
forecast_base_bill_ele   forecast_bill_12m          0.794776
cons_12m                 cons_last_month            0.919545
forecast_base_bill_ele   forecast_cons              0.964402
forecast_cons_year       imp_cons                   0.981732
cons_12m                 cons_12m                        NaN
dtype: float64
```

In [46]:
```
names = []

for n in data.index:
    if n[0] not in names:
        names.append(n[0])
    if n[1] not in names:
        names.append(n[1])

names.append("churn")
print("List of usable features : ",names)
```

```
List of usable features :  ['forecast_meter_rent_12m', 'pow_max', 'forecast_price_en
ergy_p2', 'forecast_cons_12m', 'imp_cons', 'forecast_cons_year', 'forecast_bill_12
m', 'forecast_cons', 'margin_gross_pow_ele', 'margin_net_pow_ele', 'net_margin', 'fo
recast_base_bill_ele', 'cons_12m', 'cons_last_month', 'churn']
```

Sub-task 2: Build the features in order to get ready to model

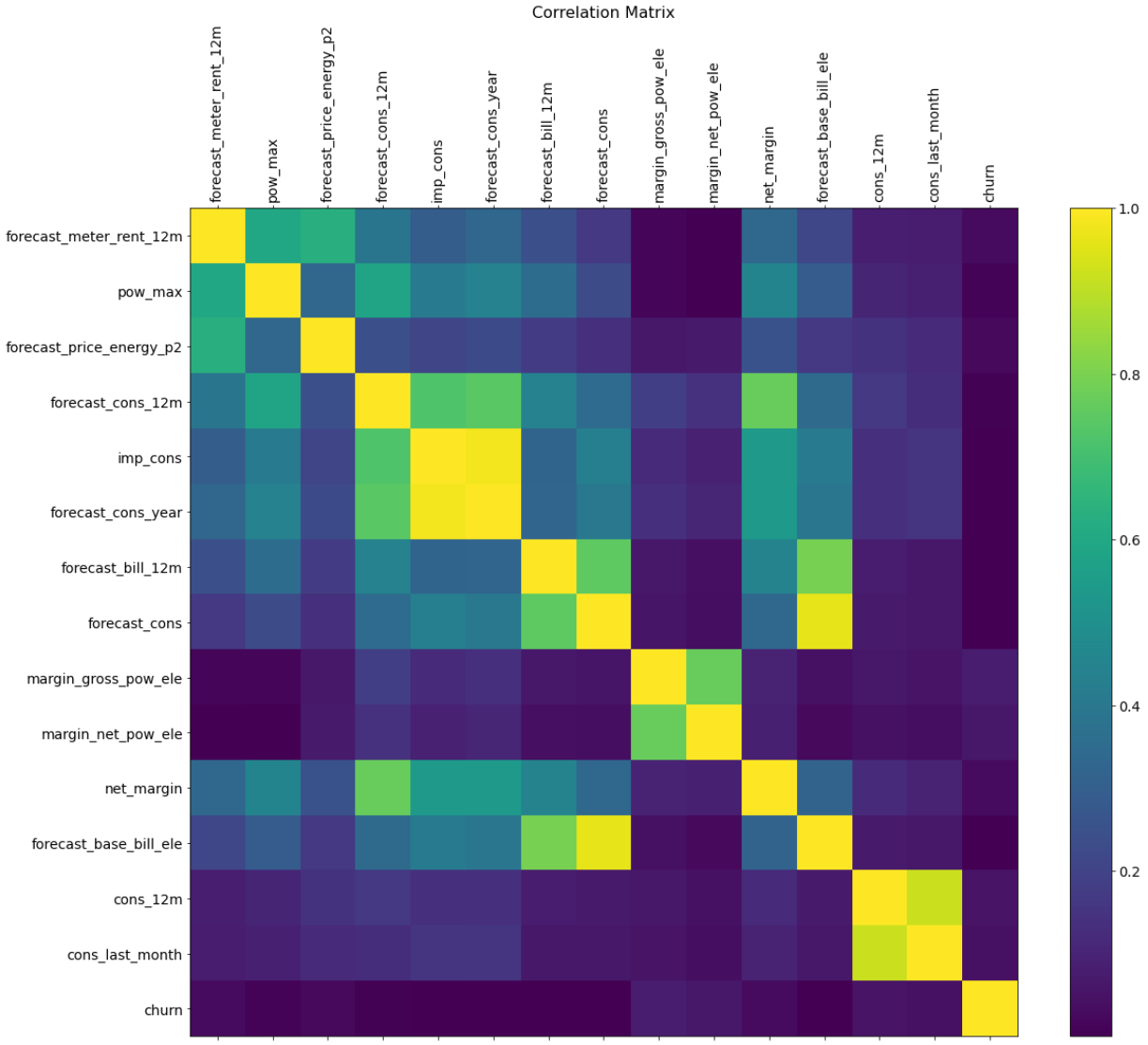
In [47]:
```
ml_trX=ml_tr[names]
```

In [48]:
```
ml_trX.head()
```

Out[48]:

	forecast_meter_rent_12m	pow_max	forecast_price_energy_p2	forecast_cons_12m	imp_cons	forec
0	359.29	180.000	0.088347	26520.30	831.8	
1	1.78	43.648	0.098142	0.00	0.0	
2	16.27	13.800	0.000000	189.95	0.0	
3	38.72	13.856	0.087899	47.96	0.0	
4	19.83	13.200	0.000000	240.04	0.0	

In [49]:
```
df=ml_trX.copy()
f = plt.figure(figsize=(19, 15))
plt.matshow(df.corr().abs(), fignum=f.number)
plt.xticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
plt.yticks(range(df.select_dtypes(['number']).shape[1]), df.select_dtypes(['number']
cb = plt.colorbar()
cb.ax.tick_params(labelsize=14)
plt.title('Correlation Matrix', fontsize=16);
```



Modeling

In [50]:
```
ml_tr['churn']
```

Out[50]:
```
0        0
1        1
2        0
3        0
4        0
        ..
16091    0
16092    1
16093    1
16094    0
16095    0
Name: churn, Length: 16096, dtype: int64
```

In [51]:
```
churn=ml_tr['churn']
ml_tr.shape
```

Out[51]:
```
(16096, 32)
```

In [56]:
```
import pycaret
from pycaret.classification import setup
from pycaret.classification import compare_models
from pycaret.classification import tune_model


grid = setup(data=ml_trX, target="churn", html=False, silent=True, verbose=False)
# evaluate models and compare models
best = compare_models()
# report the best model
print(best)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
ada	Ada Boost Classifier	0.8998	0.6532	0.0053	0.3367	0.0104	0.0072	0.0306	0.189
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
ada	Ada Boost Classifier	0.8998	0.6532	0.0053	0.3367	0.0104	0.0072	0.0306	0.189
gbc	Gradient Boosting Classifier	0.8996	0.6715	0.0134	0.3342	0.0255	0.0191	0.0511	0.718
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
lda	Linear Discriminant Analysis	0.8999	0.6125	0.0071	0.2750	0.0139	0.0101	0.0341	0.015
ada	Ada Boost Classifier	0.8998	0.6532	0.0053	0.3367	0.0104	0.0072	0.0306	0.189
gbc	Gradient Boosting Classifier	0.8996	0.6715	0.0134	0.3342	0.0255	0.0191	0.0511	0.718
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
lda	Linear Discriminant Analysis	0.8999	0.6125	0.0071	0.2750	0.0139	0.0101	0.0341	0.015
ada	Ada Boost Classifier	0.8998	0.6532	0.0053	0.3367	0.0104	0.0072	0.0306	0.189
gbc	Gradient Boosting Classifier	0.8996	0.6715	0.0134	0.3342	0.0255	0.0191	0.0511	0.718
et	Extra Trees Classifier	0.8996	0.6466	0.0713	0.4751	0.1235	0.1002	0.1537	0.320
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
lightgbm	Light Gradient Boosting Machine	0.9009	0.6659	0.0428	0.5326	0.0784	0.0647	0.1277	0.111
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019
lda	Linear Discriminant Analysis	0.8999	0.6125	0.0071	0.2750	0.0139	0.0101	0.0341	0.015
ada	Ada Boost Classifier	0.8998	0.6532	0.0053	0.3367	0.0104	0.0072	0.0306	0.189
gbc	Gradient Boosting Classifier	0.8996	0.6715	0.0134	0.3342	0.0255	0.0191	0.0511	0.718
et	Extra Trees Classifier	0.8996	0.6466	0.0713	0.4751	0.1235	0.1002	0.1537	0.320
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.9015	0.6569	0.0427	0.5871	0.0793	0.0664	0.1373	0.597
lightgbm	Light Gradient Boosting Machine	0.9009	0.6659	0.0428	0.5326	0.0784	0.0647	0.1277	0.111
ridge	Ridge Classifier	0.9003	0.0000	0.0009	0.1000	0.0018	0.0012	0.0070	0.019

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lda	Linear Discriminant Analysis	0.8999	0.6125	0.0071	0.2750	0.0139	0.0101	0.0341	0.015
ada	Ada Boost Classifier	0.8998	0.6532	0.0053	0.3367	0.0104	0.0072	0.0306	0.189
gbc	Gradient Boosting Classifier	0.8996	0.6715	0.0134	0.3342	0.0255	0.0191	0.0511	0.718
et	Extra Trees Classifier	0.8996	0.6466	0.0713	0.4751	0.1235	0.1002	0.1537	0.320
lr	Logistic Regression	0.8971	0.5409	0.0044	0.0983	0.0085	0.0005	0.0007	0.103
knn	K Neighbors Classifier	0.8971	0.5638	0.0757	0.4091	0.1273	0.0996	0.1412	0.042
svm	SVM - Linear Kernel	0.8842	0.0000	0.0268	0.1714	0.0355	0.0087	0.0190	0.020
qda	Quadratic Discriminant Analysis	0.8630	0.5595	0.0767	0.1571	0.0994	0.0354	0.0396	0.022
nb	Naive Bayes	0.8572	0.6023	0.0927	0.1508	0.1141	0.0415	0.0432	0.018
dt	Decision Tree Classifier	0.8254	0.5427	0.1898	0.1670	0.1774	0.0803	0.0806	0.057

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=-1, oob_score=False, random_state=2170, verbose=0,
                       warm_start=False)
```

best model >> Light Gradient Boosting Machine

Accuracy >> 90%

In [ ]:

In [ ]: