```
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\numpy\_distributor_init.py:32: UserWarning: loaded more than 1 DLL from .libs:
C:\Users\lucia\Anaconda3\lib\site-packages\numpy\.libs\libopenblas.PYQHXLVVQ7VESDPUVUADXEVJOBGHJPAY.gfortran-win_amd64.dll
C:\Users\lucia\Anaconda3\lib\site-packages\numpy\.libs\libopenblas.TXA6YQSD3GCQQC22GEQ54J2UDCXDXHWN.gfortran-win_amd64.dll
  stacklevel=1)
```

```
diabetis_total= pd.read_csv('TrainingWiDS2021.csv')
diabetis_NotLabeled= pd.read_csv('UnlabeledWiDS2021.csv')
Data_dictionary= pd.read_csv('DataDictionaryWiDS2021.csv')
```

```
diabetis_total=diabetis_total.drop('Unnamed: 0', axis=1) #drop column
diabetis_NotLabeled=diabetis_NotLabeled.drop('Unnamed: 0', axis=1)
```

```
diabetis, diabetis_test = train_test_split(diabetis_total, test_size=0.30, random_state=42) #split test with 30%
```

```
diabetis
```

	encounter_id	hospital_id	age	bmi	elective_surgery	ethnicity	gender	height	hospital_admit_source	icu_
91852	253264	195	61.0	42.581313	0	African American	M	172.70	Emergency Department	
44993	195113	132	20.0	31.514647	0	Caucasian	F	147.00	Emergency Department	
129445	170014	170	NaN	21.026959	0	Caucasian	F	157.50	Emergency Department	
55299	207843	194	73.0	21.387755	0	Caucasian	M	175.00	NaN	Op
102655	199415	28	71.0	31.838229	0	Other/Unknown	M	180.30	NaN	
101835	266989	110	33.0	48.059268	0	Caucasian	F	165.10	Direct Admit	
70820	199460	185	61.0	26.297275	1	Caucasian	M	195.59	NaN	Op
124784	202806	163	69.0	23.422513	1	Caucasian	F	154.90	NaN	Op
4630	218986	46	77.0	33.591638	0	Caucasian	M	180.30	Emergency Department	
27642	196284	79	88.0	24.919982	1	Asian	M	162.00	NaN	Op
23732	167759	171	76.0	22.637719	1	Caucasian	M	170.10	Operating Room	Op
58707	201242	176	74.0	26.621703	1	Caucasian	M	172.70	Emergency Department	Op
114763	208493	86	58.0	27.464937	1	Caucasian	M	175.30	Operating Room	Op
95966	248406	191	40.0	26.344120	1	Caucasian	M	167.60	NaN	Op

103129	157988	28	74.0	28.218695	0	Hispanic	M	157.50	NaN	
16573	275357	116	88.0	17.148438	0	Caucasian	F	160.00	NaN	
19466	236951	161	78.0	23.940345	1	Caucasian	M	182.00	Operating Room	Op
43460	169361	100	52.0	30.759870	0	Caucasian	M	172.00	Emergency Department	
93035	173468	203	NaN	24.341758	1	Caucasian	F	157.00	Floor	Op
76849	261625	196	77.0	37.243130	0	Caucasian	F	152.40	Emergency Department	
79013	212981	18	48.0	35.491908	0	Other/Unknown	M	177.80	Direct Admit	
33344	186201	10	78.0	25.575439	0	Caucasian	F	170.10	Emergency Department	
99504	249421	191	NaN	33.108150	0	Caucasian	F	167.60	Emergency Department	
56798	261767	187	83.0	14.844926	0	Caucasian	F	165.10	Emergency Department	
897	252080	69	88.0	21.733061	0	Caucasian	F	165.10	Emergency Department	
23356	173842	147	83.0	27.325773	0	Caucasian	F	156.00	Emergency Department	
24257	188983	79	70.0	30.761246	0	Caucasian	M	170.00	Emergency Department	
128883	187999	7	80.0	28.622530	0	Caucasian	M	167.60	Other Hospital	
126796	168905	7	39.0	37.095369	0	Other/Unknown	M	167.60	Emergency Department	
42657	193840	14	73.0	36.506220	0	Caucasian	M	172.00	Emergency Department	

...	...	...	...	...	...	...	...	...	...	
103355	211564	156	88.0	25.480824	0	Caucasian	F	163.00	Emergency Department	
5311	165209	118	29.0	23.232500	1	Caucasian	M	170.20	Operating Room	Op
67969	213383	92	75.0	16.124969	0	Caucasian	F	157.50	NaN	
121637	173968	175	30.0	44.712969	0	Native American	F	167.00	Emergency Department	
64925	243404	94	68.0	32.632674	0	Caucasian	F	155.00	Emergency Department	
62955	212127	9	81.0	45.310000	0	Caucasian	F	170.10	Floor	
59735	261545	90	69.0	29.451893	0	Caucasian	F	165.10	Emergency Department	
769	251496	77	55.0	28.849037	0	African American	M	177.80	Floor	
64820	252052	19	49.0	32.945645	0	Caucasian	M	180.30	Floor	
67221	265784	139	21.0	31.354080	0	Caucasian	M	187.90	NaN	
41090	245760	100	68.0	18.113754	0	Caucasian	M	182.00	Emergency Department	
16023	188990	98	81.0	NaN	1	Other/Unknown	M	157.48	Recovery Room	Op
60263	162249	176	77.0	25.219581	1	Caucasian	M	175.30	Recovery Room	Op
44131	274394	100	34.0	38.750078	0	Caucasian	M	177.80	Step-Down Unit (SDU)	
126324	221320	122	74.0	32.858021	0	Caucasian	M	172.70	NaN	

112727	160797	86	57.0	27.180900	0	Caucasian	F	165.00	Other Hospital	
87498	173433	35	59.0	21.249531	0	Caucasian	M	175.30	NaN	
37194	225090	70	79.0	18.810756	1	Caucasian	M	178.00	Operating Room	Op
128214	195463	7	88.0	23.332591	0	African American	F	165.10	Emergency Department	
82386	228583	18	23.0	31.933345	0	Caucasian	M	167.60	NaN	
6265	262691	118	75.0	NaN	0	Caucasian	M	182.90	Floor	
54886	179045	194	28.0	32.049982	0	African American	M	157.00	Emergency Department	
76820	232318	18	57.0	17.098918	0	Caucasian	M	182.90	PACU	Op

```
len(diabetis_NotLabeled[diabetis_NotLabeled.elective_surgery==1]) # ca 20% are elective
```

```
2048
```

```
len(diabetis_total[(diabetis_total.elective_surgery==1) & (diabetis_total.diabetes_mellitus==1) ]) # ca 20% are elective
```

```
5241
```

```
len(diabetis_total[(diabetis_total.elective_surgery==0) & (diabetis_total.diabetes_mellitus==1) ]) # ca 20% are elective
```

```
22910
```

15795	266109	116	63.0	27.447725	0	Hispanic	M	167.60	NaN

```
len(diabetis_total[(diabetis_total.elective_surgery==1)]) #25% ELECTIVE from those 20% diabetic
```

```
24709
```

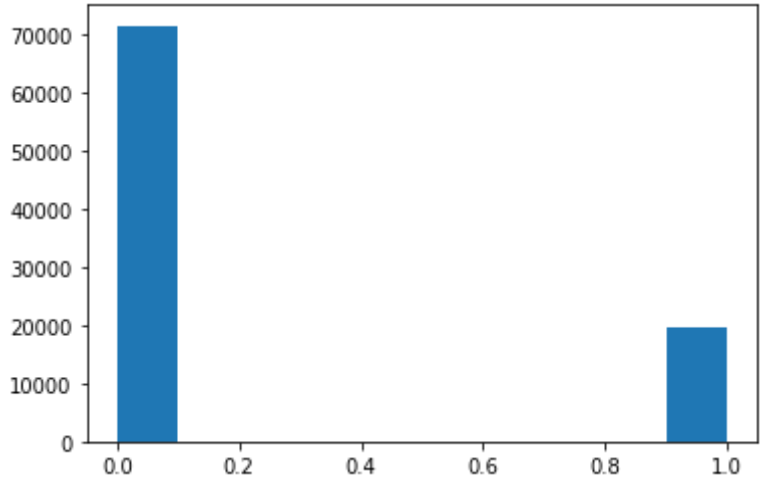
```
len(diabetis_NotLabeled[(diabetis_NotLabeled.elective_surgery==1)])
```

```
2048
```

```
import matplotlib.pyplot as plt
```

```
plt.hist(diabetis.diabetes_mellitus) # positive and negative counts
```

```
(array([71432.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
            0., 19677.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



LightGBM model

```
diabetis_total.info() # gather information about the types of values available
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 130157 entries, 0 to 130156
Columns: 180 entries, encounter_id to diabetes_mellitus
dtypes: float64(157), int64(17), object(6)
memory usage: 178.7+ MB
```

```
#transform object variables into categgory type for LGBM
for c in diabetis_total.columns:
    col_type = diabetis_total[c].dtype
    if col_type == 'object':
        diabetis_total[c] = diabetis_total[c].astype('category')
```

```
diabetis_total.info() # to confirm
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 130157 entries, 0 to 130156
Columns: 180 entries, encounter_id to diabetes_mellitus
dtypes: category(6), float64(157), int64(17)
memory usage: 173.5 MB
```

```
y_train=diabetis['diabetes_mellitus']
X_train=diabetis.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status'], axis=1 )
y_test=diabetis_test['diabetes_mellitus']
X_test=diabetis_test.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status'], axis=1 )
```

```
#transform object variables into categgory type for LGBM
for c in X_train.columns:
    col_type = X_train[c].dtype
    if col_type == 'object':
        X_train[c] = X_train[c].astype('category')
```

```
#transform object variables into categgory type for LGBM
for c in X_test.columns:
    col_type = X_test[c].dtype
    if col_type == 'object':
        X_test[c] = X_test[c].astype('category')
```

```
import lightgbm as lgb #pip install lightgbm
#kind of optimized model
clf_2 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary', n_jobs=-1,
                              metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                             learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                             n_estimators=5000, max_depth=-1, num_leaves=100 )


#fit the lghbm model
clf_2.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.826671
[200]   test's auc: 0.834565
[300]   test's auc: 0.839439
[400]   test's auc: 0.843873
[500]   test's auc: 0.847121
[600]   test's auc: 0.849906
[700]   test's auc: 0.851809
[800]   test's auc: 0.853489
[900]   test's auc: 0.854612
[1000]  test's auc: 0.855602
[1100]  test's auc: 0.856421
[1200]  test's auc: 0.85715
[1300]  test's auc: 0.85772
[1400]  test's auc: 0.858218
[1500]  test's auc: 0.858718
[1600]  test's auc: 0.859177
[1700]  test's auc: 0.859399
[1800]  test's auc: 0.859649
[1900]  test's auc: 0.859859
[2000]  test's auc: 0.860082
[2100]  test's auc: 0.860248
[2200]  test's auc: 0.860408
[2300]  test's auc: 0.860526
[2400]  test's auc: 0.860626
[2500]  test's auc: 0.860643
[2600]  test's auc: 0.860728
```

```
[2700]  test's auc: 0.86079
[2800]  test's auc: 0.860768
[2900]  test's auc: 0.860839
[3000]  test's auc: 0.860853
[3100]  test's auc: 0.860908
[3200]  test's auc: 0.860919
[3300]  test's auc: 0.86098
[3400]  test's auc: 0.861002
[3500]  test's auc: 0.861019
[3600]  test's auc: 0.861032
[3700]  test's auc: 0.861013
[3800]  test's auc: 0.860974
Early stopping, best iteration is:
[3584]  test's auc: 0.861041
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=10000, num_leaves=100,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
#what about droping elective surgeries, they know information about them!!
#diabetis2=diabetis.drop(diabetis[diabetis.elective_surgery==1].index, axis=0)
#diabetis_test2=diabetis_test.drop(diabetis_test[diabetis_test.elective_surgery==1].index, axis=0)
y_train2=diabetis2['diabetes_mellitus']
X_train2=diabetis2.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status', 'elective_surgery'], axis
y_test2=diabetis_test2['diabetes_mellitus']
X_test2=diabetis_test2.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status', 'elective_surgery'],


#transform object variables into categgory type for LGBM
for c in X_train2.columns:
    col_type = X_train2[c].dtype
    if col_type == 'object':
        X_train2[c] = X_train2[c].astype('category')

#transform object variables into categgory type for LGBM
for c in X_test2.columns:
    col_type = X_test2[c].dtype
    if col_type == 'object':
        X_test2[c] = X_test2[c].astype('category')
```

```
import lightgbm as lgb #pip install lightgbm
#kind of optimized model
clf_2 = lgb.LGBMClassifier(boosting_type='gbdt', objective='cross_entropy', n_jobs=-1,
                               metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                              learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                              n_estimators = 10000, max_depth=-1, num_leaves=50) # 0.85576


#fit the lghbm model
clf_2.fit(X_train2, y_train2, early_stopping_rounds= 500, eval_metric= 'auc',
                            eval_set= [(X_test2,y_test2)], eval_names= ['test'], verbose= 500, feature_name= 'auto', categorical_feat
```

```
Training until validation scores don't improve for 500 rounds
[500]   test's auc: 0.852356
[1000]  test's auc: 0.860315
[1500]  test's auc: 0.863255
[2000]  test's auc: 0.864261
[2500]  test's auc: 0.864884
[3000]  test's auc: 0.865216
[3500]  test's auc: 0.865433
[4000]  test's auc: 0.865546
[4500]  test's auc: 0.865634
[5000]  test's auc: 0.865674
Early stopping, best iteration is:
[4793]  test's auc: 0.865708
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=10000, num_leaves=50,
               objective='cross_entropy', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
#fit the model in the full labeled dataset
diabetis_total2=diabetis_total.drop(diabetis_total[diabetis_total.elective_surgery==1].index, axis=0)

y_LGBM2=diabetis_total2['diabetes_mellitus']
X_LGBM2=diabetis_total2.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status', 'elective_surgery'],

#transform object variables into category type
```

```
#transform object variables into category type
for c in X_LGBM2.columns:
    col_type = X_LGBM2[c].dtype
    if col_type == 'object':
        X_LGBM2[c] = X_LGBM2[c].astype('category')

clf_2 = lgb.LGBMClassifier(boosting_type='gbdt', objective='cross_entropy', n_jobs=-1,
                               metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                              learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                              n_estimators = 5000, max_depth=-1, num_leaves=50) # 0.85576

clf_2.fit(X_LGBM2, y_LGBM2)
```

```
     LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
                    metric='auc', n_estimators=5000, num_leaves=50,
                    objective='cross_entropy', reg_alpha=3, reg_lambda=1,
                    scale_pos_weight=0.5, subsample=1)
```

```
#transform object variables into catgegory type in the not labeled dataset
for c in diabetis_NotLabeled.columns:
    col_type = diabetis_NotLabeled[c].dtype
    if col_type == 'object':
        diabetis_NotLabeled[c] = diabetis_NotLabeled[c].astype('category')
X_Notlabeled= diabetis_NotLabeled.drop(['encounter_id', 'hospital_id', 'icu_id', 'readmission_status', 'elective_surgery'], axis=1 )


Yhat_LGBM_optimiz_prob_2=clf_2.predict_proba(X_Notlabeled)[:,1] # predict probabilities for the positive class after fitting the full


# submission data
solution_temp = pd.read_csv('SolutionTemplateWiDS2021.csv')
test_encounter_id=diabetis_NotLabeled['encounter_id']
test_pred_map = dict(zip(test_encounter_id, Yhat_LGBM_optimiz_prob_2))
solution_temp['diabetes_mellitus'] = solution_temp['encounter_id'].map(test_pred_map)
solution_temp.to_csv('submission_df13.csv', index = False)
# score 0.85718 ! The best ever we have had df_11
# score 0.85636 df_12
# score 0.85455 df_13 # ends up being worse anyway!!
```

```
#what about learning only from the elective surgeries, they indeed know previously if the person is diabetic or not...


#what about droping elective surgeries, they know information about them!!
diabetis3=diabetis.drop(diabetis[diabetis.elective_surgery==0].index, axis=0)
diabetis_test3=diabetis_test.drop(diabetis_test[diabetis_test.elective_surgery==0].index, axis=0)
y_train3=diabetis3['diabetes_mellitus']
X_train3=diabetis3.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status', 'elective_surgery'], axis
y_test3=diabetis_test3['diabetes_mellitus']
X_test3=diabetis_test3.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status', 'elective_surgery'],


#transform object variables into categgory type for LGBM
for c in X_train3.columns:
    col_type = X_train3[c].dtype
    if col_type == 'object':
        X_train3[c] = X_train3[c].astype('category')

#transform object variables into categgory type for LGBM
for c in X_test3.columns:
    col_type = X_test3[c].dtype
    if col_type == 'object':
        X_test3[c] = X_test3[c].astype('category')


#fit the lghbm model
clf_2.fit(X_train3, y_train3, early_stopping_rounds= 500, eval_metric= 'auc',
                            eval_set= [(X_test3,y_test3)], eval_names= ['test'], verbose= 500, feature_name= 'auto', categorical_feat
```

```
Training until validation scores don't improve for 500 rounds
[500]   test's auc: 0.822245
[1000]  test's auc: 0.827559
[1500]  test's auc: 0.828617
[2000]  test's auc: 0.828396
Early stopping, best iteration is:
[1518]  test's auc: 0.828687
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=10000, num_leaves=50,
```

```
                    objective='cross_entropy', reg_alpha=3, reg_lambda=1,
                    scale_pos_weight=0.5, subsample=1)
```

```
# drop columns with more than 75% NA
columns_to_drop=X_train.isna().sum()[X_train.isna().sum() > len(X_train)*0.75] #columns with more than 75% NA
```

```
columns_to_drop
```

```
     fio2_apache               69828
     paco2_apache              69828
     paco2_for_ph_apache       69828
     pao2_apache               69828
     ph_apache                 69828
     h1_diasbp_invasive_max    73413
     h1_diasbp_invasive_min    73413
     h1_mbp_invasive_max       73372
     h1_mbp_invasive_min       73372
     h1_sysbp_invasive_max     73394
     h1_sysbp_invasive_min     73394
     h1_albumin_max            83346
     h1_albumin_min            83346
     h1_bilirubin_max          83945
     h1_bilirubin_min          83945
     h1_bun_max                73515
     h1_bun_min                73515
     h1_calcium_max            74172
     h1_calcium_min            74172
     h1_creatinine_max         73395
     h1_creatinine_min         73395
     h1_hco3_max               74523
     h1_hco3_min               74523
     h1_hemaglobin_max         72008
     h1_hemaglobin_min         72008
     h1_hematocrit_max         72115
     h1_hematocrit_min         72115
     h1_lactate_max            82956
     h1_lactate_min            82956
     h1_platelets_max          74067
```

```
h1_platelets_min          74067
h1_potassium_max          70621
h1_potassium_min          70621
h1_sodium_max             71292
h1_sodium_min             71292
h1_wbc_max                74242
h1_wbc_min                74242
h1_arterial_pco2_max      75379
h1_arterial_pco2_min      75379
h1_arterial_ph_max        75539
h1_arterial_ph_min        75539
h1_arterial_po2_max       75237
h1_arterial_po2_min       75237
h1_pao2fio2ratio_max      79378
h1_pao2fio2ratio_min      79378
dtype: int64
```

```
X_train=X_train.drop(columns_to_drop.index.to_list(), axis=1)
X_test=X_test.drop(columns_to_drop.index.to_list(), axis=1)


#imput the mean in NAN
selection_main_=['bun','albumin', 'creatinine',  'lactate', 'bilirubin','hematocrit', 'hemaglobin', 'sysbp',
                    'mbp', 'diasbp','sodium', 'potassium', 'hco3', 'glucose', 'pco2', 'po2', 'calcium',
                    'heartrate', 'inr', 'pao2fio2ratio', 'platelets', 'resprate', 'spo2', 'temp', 'wbc']

for word in selection_main_:
    variables=X_train.columns[X_train.columns.str.contains(word)==True]
    average1=X_train[variables].mean(axis=1)
    average2=X_test[variables].mean(axis=1)
    X_train[variables]= X_train[variables].T.fillna(average1).T
    X_test[variables]= X_test[variables].T.fillna(average2).T


#imput ag, bmi, weight and height
#use the mean or median height, weight, age when there is no information
X_train['height'] = X_train['height'].fillna(X_train.groupby("gender")["height"].transform("mean"))
```

```
X_train['height'] = X_train['height'].fillna(X_train.groupby("gender")["height"].transform("mean"))
X_train['weight'] = X_train['weight'].fillna(X_train.groupby("gender")["weight"].transform("median"))
X_train['age'] = X_train['age'].fillna(X_train.groupby("gender")["age"].transform("median"))

#the same for test
X_test['height'] = X_test['height'].fillna(X_test.groupby("gender")["height"].transform("mean"))
X_test['weight'] = X_test['weight'].fillna(X_test.groupby("gender")["weight"].transform("median"))
X_test['age'] = X_test['age'].fillna(X_test.groupby("gender")["age"].transform("median"))

#bmi seems important
#calculate bmi from weight and height, formula= weight/(height**2)
X_train['bmi']=X_train['bmi'].fillna(X_train['weight']/((X_train['height']/100)**2))
X_test['bmi']=X_test['bmi'].fillna(X_test['weight']/((X_test['height']/100)**2))


#import lightgbm as lgb #pip install lightgbm
#kind of optimized model
clf_2 = lgb.LGBMClassifier(boosting_type='gbdt', objective='cross_entropy', n_jobs=-1,
                             metric= 'auc', scale_pos_weight= 5, subsample = 0.8, force_col_wise = True,
                            learning_rate= 0.01, colsample_bytree= 0.5, reg_alpha= 3, reg_lambda= 1,
                            n_estimators = 100000, max_depth=-1, num_leaves=2)


#fit the lghbm model
clf_2.fit(X_train, y_train, early_stopping_rounds= 500, eval_metric= 'auc',
                           eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 10000, feature_name= 'auto', categorical_feat
```

```
Training until validation scores don't improve for 500 rounds
[10000] test's auc: 0.842207
[20000] test's auc: 0.845166
[30000] test's auc: 0.846119
[40000] test's auc: 0.84654
[50000] test's auc: 0.846725
[60000] test's auc: 0.846818
Early stopping, best iteration is:
[59562] test's auc: 0.84682
LGBMClassifier(colsample_bytree=0.5, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=100000, num_leaves=2,
               objective='cross_entropy', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=5, subsample=0.8)
```

```
#fit the model in the full labeled dataset

y_LGBM=diabetis_total['diabetes_mellitus']
X_LGBM=diabetis_total.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id', 'readmission_status'], axis=1 )

#transform object variables into catgegory type
for c in X_LGBM.columns:
    col_type = X_LGBM[c].dtype
    if col_type == 'object':
        X_LGBM[c] = X_LGBM[c].astype('category')

clf_2 = lgb.LGBMClassifier(boosting_type='gbdt', objective='cross_entropy', n_jobs=-1,
                               metric= 'auc', scale_pos_weight= 1, subsample = 1, force_col_wise = True,
                              learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                              n_estimators = 4203, max_depth=-1, num_leaves=50) # 0.85576

clf_2.fit(X_LGBM, y_LGBM)

     LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
                    metric='auc', n_estimators=4203, num_leaves=50,
                    objective='cross_entropy', reg_alpha=3, reg_lambda=1,
                    scale_pos_weight=1, subsample=1)


#transform object variables into catgegory type in the not labeled dataset
for c in diabetis_NotLabeled.columns:
    col_type = diabetis_NotLabeled[c].dtype
    if col_type == 'object':
        diabetis_NotLabeled[c] = diabetis_NotLabeled[c].astype('category')
X_Notlabeled= diabetis_NotLabeled.drop(['encounter_id', 'hospital_id', 'icu_id', 'readmission_status'], axis=1 ) #features not labele


Yhat_LGBM_optimiz_prob_=clf_2.predict_proba(X_Notlabeled)[:,1] # predict probabilities for the positive class after fitting the full


# submission data
solution_temp = pd.read_csv('SolutionTemplateWiDS2021.csv')
test_encounter_id=diabetis_NotLabeled['encounter_id']
```

```
test_pred_map = dict(zip(test_encounter_id, Yhat_LGBM_optimiz_prob_))
solution_temp['diabetes_mellitus'] = solution_temp['encounter_id'].map(test_pred_map)
solution_temp.to_csv('submission_df12.csv', index = False)
# score 0.85718 ! The best ever we have had df_11
# score 0.85636 df_12


#changed number of leaves to 50
clf_2_ = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                              metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                             learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                             n_estimators=5000, max_depth=-1, num_leaves=50 )


#fit the lghbm model
clf_2_.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.823784
[200]   test's auc: 0.832009
[300]   test's auc: 0.837067
[400]   test's auc: 0.841789
[500]   test's auc: 0.845213
[600]   test's auc: 0.848101
[700]   test's auc: 0.850119
[800]   test's auc: 0.851792
[900]   test's auc: 0.85305
[1000]  test's auc: 0.854226
[1100]  test's auc: 0.855174
[1200]  test's auc: 0.855961
[1300]  test's auc: 0.856613
[1400]  test's auc: 0.857218
[1500]  test's auc: 0.857809
[1600]  test's auc: 0.858263
[1700]  test's auc: 0.858572
[1800]  test's auc: 0.858812
[1900]  test's auc: 0.859108
[2000]  test's auc: 0.85936
[2100]  test's auc: 0.859526
```

```
[2200]	test's auc: 0.859705
[2300]	test's auc: 0.859841
[2400]	test's auc: 0.859995
[2500]	test's auc: 0.860075
[2600]	test's auc: 0.860191
[2700]	test's auc: 0.860239
[2800]	test's auc: 0.860288
[2900]	test's auc: 0.860387
[3000]	test's auc: 0.860423
[3100]	test's auc: 0.860475
[3200]	test's auc: 0.860567
[3300]	test's auc: 0.860612
[3400]	test's auc: 0.860627
[3500]	test's auc: 0.860693
[3600]	test's auc: 0.860731
[3700]	test's auc: 0.86077
[3800]	test's auc: 0.860797
[3900]	test's auc: 0.860819
[4000]	test's auc: 0.860855
[4100]	test's auc: 0.860911
[4200]	test's auc: 0.860923
[4300]	test's auc: 0.860974
[4400]	test's auc: 0.861004
[4500]	test's auc: 0.860995
[4600]	test's auc: 0.861001
[4700]	test's auc: 0.860997
[4800]	test's auc: 0.861023
[4900]	test's auc: 0.861025
[5000]	test's auc: 0.861012
Did not meet early stopping. Best iteration is:
[4891]	test's auc: 0.861037
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=50,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
#transform object variables into catgegory type in the not labeled dataset
for c in diabetis_NotLabeled.columns:
    col_type = diabetis_NotLabeled[c].dtype
    if col_type == 'object':
        diabetis_NotLabeled[c] = diabetis_NotLabeled[c].astype('category')
```

```
        diabetis_NotLabeled[c] = diabetis_NotLabeled[c].astype('category')
X_Notlabeled= diabetis_NotLabeled.drop(['encounter_id', 'hospital_id', 'icu_id'], axis=1 ) #features not labeled dataset


Yhat_LGBM_optimiz_prob_FitOnlyTrain=clf_2.predict_proba(X_Notlabeled)[:,1] # predict probabilities for the positive class


# submission data
solution_temp = pd.read_csv('SolutionTemplateWiDS2021.csv')
test_encounter_id=diabetis_NotLabeled['encounter_id']
test_pred_map = dict(zip(test_encounter_id, Yhat_LGBM_optimiz_prob_FitOnlyTrain))
solution_temp['diabetes_mellitus'] = solution_temp['encounter_id'].map(test_pred_map)
solution_temp.to_csv('submission_df9.csv', index = False)
# score 0.85334


#fit the model in the full labeled dataset

y_LGBM=diabetis_total['diabetes_mellitus']
X_LGBM=diabetis_total.drop(['diabetes_mellitus','encounter_id', 'hospital_id', 'icu_id'], axis=1 )

#transform object variables into catgegory type
for c in X_LGBM.columns:
    col_type = X_LGBM[c].dtype
    if col_type == 'object':
        X_LGBM[c] = X_LGBM[c].astype('category')

clf_2.fit(X_LGBM, y_LGBM)

     LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
                    metric='auc', n_estimators=5000, num_leaves=100,
                    objective='binary', reg_alpha=3, reg_lambda=1,
                    scale_pos_weight=0.5, subsample=1)


Yhat_LGBM_optimiz_prob=clf_2.predict_proba(X_Notlabeled)[:,1] # predict probabilities for the positive class after fitting the full d


# submission data
solution_temp = pd.read_csv('SolutionTemplateWiDS2021.csv')
test_encounter_id=diabetis_NotLabeled['encounter_id']
```

```
test_encounter_id=diabetis_NotLabeled[ encounter_id ]
test_pred_map = dict(zip(test_encounter_id, Yhat_LGBM_optimiz_prob))
solution_temp['diabetes_mellitus'] = solution_temp['encounter_id'].map(test_pred_map)
solution_temp.to_csv('submission_df10.csv', index = False)
# score 0.85694 ! The best ever we have had
```

using the relevant ones with 5% threshold previously analyzed

```
relevant= ['age','bmi','pre_icu_los_days','weight','apache_post_operative','arf_apache','bilirubin_apache','bun_apache','creatinine_a
            'gcs_unable_apache','glucose_apache','pao2_apache','d1_diasbp_invasive_min','h1_diasbp_invasive_min','d1_bilirubin_max','
            'd1_bun_max','d1_bun_min','d1_creatinine_max','d1_creatinine_min','d1_glucose_max','d1_glucose_min','h1_bilirubin_max',
            'h1_bilirubin_min','h1_bun_max','h1_bun_min','h1_creatinine_max','h1_creatinine_min','h1_glucose_max','h1_glucose_min',
            'd1_arterial_po2_min','h1_arterial_po2_min','aids','cirrhosis','hepatic_failure','immunosuppression','leukemia','lymphoma
            'solid_tumor_with_metastasis', 'icu_type', 'ethnicity', 'gender'] #the releveant 5% threshold + ethnicity+ gender and icu
```

```
import lightgbm as lgb
X_train2=X_train[relevant]
X_test2=X_test[relevant]

for c in X_train2.columns:
    col_type = X_train2[c].dtype
    if col_type == 'object':
        X_train2[c] = X_train2[c].astype('category')

for c in X_test2.columns:
    col_type = X_test2[c].dtype
    if col_type == 'object':
        X_test2[c] = X_test2[c].astype('category')
```

```
clf_3 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                              metric= 'auc', scale_pos_weight= 20, subsample = 1, force_col_wise = True,
                             learning_rate= 0.001, colsample_bytree= 0.56, reg_alpha= 0, reg_lambda= 0,
                             n_estimators=10000, max_depth=-1, num_leaves=150 )
```

```
clf_3.fit(X_train2, y_train, early_stopping_rounds= 120, eval_metric= 'auc',
                            eval_set= [(X_test2,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featu
```

```
[3400]	test's auc: 0.834381
[3500]	test's auc: 0.834664
[3600]	test's auc: 0.834954
[3700]	test's auc: 0.835235
[3800]	test's auc: 0.835477
[3900]	test's auc: 0.835708
[4000]	test's auc: 0.835947
[4100]	test's auc: 0.836192
[4200]	test's auc: 0.836402
[4300]	test's auc: 0.836627
[4400]	test's auc: 0.836842
[4500]	test's auc: 0.83705
[4600]	test's auc: 0.83723
[4700]	test's auc: 0.837418
[4800]	test's auc: 0.837589
[4900]	test's auc: 0.837761
[5000]	test's auc: 0.837932
[5100]	test's auc: 0.838102
[5200]	test's auc: 0.83825
[5300]	test's auc: 0.838393
[5400]	test's auc: 0.838544
[5500]	test's auc: 0.838686
[5600]	test's auc: 0.838843
[5700]	test's auc: 0.838966
[5800]	test's auc: 0.839102
[5900]	test's auc: 0.839237
[6000]	test's auc: 0.839353
[6100]	test's auc: 0.839462
[6200]	test's auc: 0.839572
[6300]	test's auc: 0.839661
[6400]	test's auc: 0.839739
[6500]	test's auc: 0.839827
[6600]	test's auc: 0.839903
[6700]	test's auc: 0.839981
[6800]	test's auc: 0.840057
[6900]	test's auc: 0.840136
[7000]	test's auc: 0.840209
[7100]	test's auc: 0.84029
```

```
[7100]  test's auc: 0.840229
[7200]  test's auc: 0.840329
[7300]  test's auc: 0.840377
[7400]  test's auc: 0.840438
[7500]  test's auc: 0.840495
[7600]  test's auc: 0.840548
[7700]  test's auc: 0.840581
[7800]  test's auc: 0.840622
[7900]  test's auc: 0.840643
[8000]  test's auc: 0.840673
[8100]  test's auc: 0.840695
[8200]  test's auc: 0.840716
[8300]  test's auc: 0.840734
[8400]  test's auc: 0.840749
[8500]  test's auc: 0.84077
[8600]  test's auc: 0.84077
Early stopping, best iteration is:
[8484]  test's auc: 0.840772
LGBMClassifier(colsample_bytree=0.56, force_col_wise=True, learning_rate=0.001,
               metric='auc', n_estimators=10000, num_leaves=150,
               objective='binary', reg_alpha=0, reg_lambda=0,
               scale_pos_weight=20, subsample=1)
```

only the pre selected variables

```
#selection of the most relevant features for diabetes according to our more scientific search!
selection=['bun','albumin', 'creatinine',  'lactate', 'bilirubin', 'arf', 'cirrhosis', 'bmi',
           'weight', 'age', 'gender', 'ethnicity', 'glucose'] # most probable relevant features for diabetes


lista=[] # list of the relevant features
for word in selection:
    a=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    lista.append(a.to_list())

# converts to single list
flat_list = [item for sublista in lista for item in sublista]
```

```
X_train3=X_train[flat_list]
X_test3=X_test[flat_list]

for c in X_train3.columns:
    col_type = X_train3[c].dtype
    if col_type == 'object':
        X_train3[c] = X_train3[c].astype('category')

for c in X_test3.columns:
    col_type = X_test3[c].dtype
    if col_type == 'object':
        X_test3[c] = X_test3[c].astype('category')
```

```
import lightgbm as lgb #pip install lightgbm
#kind of optimized model
clf_4 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                               metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                              learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                              n_estimators=5000, max_depth=-1, num_leaves=100 )


#fit the lghbm model
clf_4.fit(X_train3, y_train, early_stopping_rounds= 120, eval_metric= 'auc',
                            eval_set= [(X_test3,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featu
```

```
Training until validation scores don't improve for 120 rounds
[100]   test's auc: 0.810371
[200]   test's auc: 0.816472
[300]   test's auc: 0.821714
[400]   test's auc: 0.82443
[500]   test's auc: 0.826949
[600]   test's auc: 0.828732
[700]   test's auc: 0.830315
[800]   test's auc: 0.831592
```

```
[900]	test's auc: 0.832628
[1000]	test's auc: 0.833484
[1100]	test's auc: 0.83411
[1200]	test's auc: 0.834533
[1300]	test's auc: 0.834857
[1400]	test's auc: 0.83515
[1500]	test's auc: 0.835359
[1600]	test's auc: 0.83555
[1700]	test's auc: 0.835732
[1800]	test's auc: 0.835828
[1900]	test's auc: 0.835935
[2000]	test's auc: 0.835994
[2100]	test's auc: 0.836026
[2200]	test's auc: 0.836009
Early stopping, best iteration is:
[2163]	test's auc: 0.836055
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=100,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

merge 5% threshold with science based ones...

```
difference=set(relevant)-set(flat_list)
merged_importances=flat_list
for x in difference:
  merged_importances.append(x)


X_train4=X_train[merged_importances]
X_test4=X_test[merged_importances]

for c in X_train4.columns:
    col_type = X_train4[c].dtype
```

```
    if col_type == 'object':
        X_train4[c] = X_train4[c].astype('category')

for c in X_test4.columns:
    col_type = X_test4[c].dtype
    if col_type == 'object':
        X_test4[c] = X_test4[c].astype('category')


#import lightgbm as lgb #pip install lightgbm
#kind of optimized model
clf_5 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                               metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                              learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                              n_estimators=5000, max_depth=-1, num_leaves=200 )


#fit the lghbm model
clf_5.fit(X_train4, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test4,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featu
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.817676
[200]   test's auc: 0.82323
[300]   test's auc: 0.828636
[400]   test's auc: 0.831861
[500]   test's auc: 0.834227
[600]   test's auc: 0.836043
[700]   test's auc: 0.837442
[800]   test's auc: 0.838439
[900]   test's auc: 0.839337
[1000]  test's auc: 0.83999
[1100]  test's auc: 0.840358
[1200]  test's auc: 0.840668
[1300]  test's auc: 0.840939
[1400]  test's auc: 0.841226
[1500]  test's auc: 0.841496
[1600]  test's auc: 0.841674
[1700]  test's auc: 0.841668
[1800]  test's auc: 0.841635
```

```
Early stopping, best iteration is:
[1614]  test's auc: 0.841697
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=200,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

Not taking glucose

```
#not taking glucose
#selection of the most relevant features for diabetes according to our more scientific search!
selection2=['bun','albumin', 'creatinine',  'lactate', 'bilirubin', 'arf', 'cirrhosis', 'bmi',
           'weight', 'age', 'gender', 'ethnicity'] # most probable relevant features for diabetes
#not taking glucose levels...


lista2=[] # list of the relevant features
for word in selection2:
    a=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    lista2.append(a.to_list())

# converts to single list
flat_list2 = [item for sublista in lista2 for item in sublista]


merged_importances2=flat_list2
for x in difference:
  merged_importances2.append(x)


X_train5=X_train[merged_importances2]
X_test5=X_test[merged_importances2]

for c in X_train5.columns:
    col_type = X_train5[c].dtype
    if col_type == 'object':
```

```
        X_train5[c] = X_train5[c].astype('category')

for c in X_test5.columns:
    col_type = X_test5[c].dtype
    if col_type == 'object':
        X_test5[c] = X_test5[c].astype('category')


#fit the lghbm model
clf_5.fit(X_train5, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test5,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featu
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.678555
[200]   test's auc: 0.688462
[300]   test's auc: 0.693967
[400]   test's auc: 0.698351
[500]   test's auc: 0.701546
[600]   test's auc: 0.703519
[700]   test's auc: 0.705116
[800]   test's auc: 0.706338
[900]   test's auc: 0.706886
[1000]  test's auc: 0.707416
[1100]  test's auc: 0.707632
[1200]  test's auc: 0.707612
Early stopping, best iteration is:
[1058]  test's auc: 0.707667
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=200,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
#without glucose levels score is much worse
```

with additional variables

```
#adding: uninoutput, hematocrit, hemaglobin, blood pressure (bp), pH, sodium, potassium, hco3
```

```
#adding: urineoutput, hematocrit, hemaglobin, blood pressure (bp), ph, sodium, potassium, hco3

selection3=['bun','albumin', 'creatinine',  'lactate', 'bilirubin', 'arf', 'cirrhosis', 'bmi',
           'weight', 'age', 'gender', 'ethnicity', 'glucose', 'urineoutput', 'hematocrit',
           'hemaglobin', 'bp', 'ph_', 'sodium', 'potassium', 'hco3']


lista3=[] # list of the relevant features
for word in selection3:
    a=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    lista3.append(a.to_list())

# converts to single list
flat_list3 = [item for sublista in lista3 for item in sublista]


len(flat_list3) # 101 variables...
```

```
101
```

```
X_train6=X_train[flat_list3]
X_test6=X_test[flat_list3]

for c in X_train6.columns:
    col_type = X_train6[c].dtype
    if col_type == 'object':
        X_train6[c] = X_train6[c].astype('category')

for c in X_test6.columns:
    col_type = X_test6[c].dtype
    if col_type == 'object':
        X_test6[c] = X_test6[c].astype('category')
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  import sys
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  if sys.path[0] == '':
```

```
X_train6.isna().sum() # a lot of missing values
```

```
bun_apache                  17912
d1_bun_max                   9664
d1_bun_min                   9664
h1_bun_max                  73515
h1_bun_min                  73515
albumin_apache              54832
d1_albumin_max              50113
d1_albumin_min              50113
h1_albumin_max              83346
h1_albumin_min              83346
creatinine_apache           17535
d1_creatinine_max            9337
d1_creatinine_min            9337
h1_creatinine_max           73395
h1_creatinine_min           73395
d1_lactate_max              66800
d1_lactate_min              66800
h1_lactate_max              82956
h1_lactate_min              82956
bilirubin_apache            57943
d1_bilirubin_max            53884
d1_bilirubin_min            53884
h1_bilirubin_max            83945
h1_bilirubin_min            83945
arf_apache                      0
cirrhosis                       0
bmi                          3158
weight                       2466
age                          3498
gender                         49
                            ...
h1_mbp_invasive_min         73372
```

```
h1_mbp_max                   4523
h1_mbp_min                   4523
h1_mbp_noninvasive_max       9258
h1_mbp_noninvasive_min       9258
h1_sysbp_invasive_max       73394
h1_sysbp_invasive_min       73394
h1_sysbp_max                 3837
h1_sysbp_min                 3837
h1_sysbp_noninvasive_max     7878
h1_sysbp_noninvasive_min     7878
paco2_for_ph_apache         69828
ph_apache                   69828
d1_arterial_ph_max          59475
d1_arterial_ph_min          59475
h1_arterial_ph_max          75539
h1_arterial_ph_min          75539
sodium_apache               17306
d1_sodium_max                9352
d1_sodium_min                9352
h1_sodium_max               71292
h1_sodium_min               71292
d1_potassium_max             8821
d1_potassium_min             8821
h1_potassium_max            70621
h1_potassium_min            70621
d1_hco3_max                 14074
d1_hco3_min                 14074
```

impute the mean in missing values

```
#glucose
glucose_variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains('glucose')==True]
glucose_mean=X_train6[glucose_variables].mean(axis=1)
```

```
55299     165.000000
102655    190.600000
101835    158.000000
70820            NaN
```

```
124784    221.400000
4630      124.800000
27642     128.000000
23732     184.666667
58707     225.666667
114763    134.800000
95966     139.000000
103129    124.200000
16573      86.000000
19466     120.666667
43460     167.000000
93035     125.200000
76849      95.666667
79013     102.000000
33344     269.600000
99504      89.000000
56798     163.000000
897        91.000000
23356     145.200000
24257     288.000000
128883    246.666667
126796           NaN
42657     230.800000
             ...
103355    143.333333
5311      110.000000
67969     227.200000
121637    107.000000
64925     130.333333
62955            NaN
59735      88.000000
769       154.800000
64820     147.400000
67221      96.200000
41090     153.200000
16023     163.000000
60263     213.000000
44131     167.000000
126324    300.400000
112727    163.000000
87498     162.666667
37194     148.800000
```

```
128214    101.666667
82386      88.666667
6265      159.400000
54886      96.000000
76820     185.800000
110268    119.000000
119879           NaN
128106    119.000000
103694    322.400000
860       143.000000
15795     168.666667
121958    183.000000
Length: 91109, dtype: float64
```

```
X_train6[glucose_variables].isna().sum()
```

```
glucose_apache    10355
d1_glucose_max     5815
d1_glucose_min     5815
h1_glucose_max    52576
h1_glucose_min    52576
dtype: int64
```

```
X_train6[glucose_variables]=X_train6[glucose_variables].T.fillna(glucose_mean).T
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\pandas\core\frame.py:3391: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self[k1] = value[k2]
```

```
X_train6[glucose_variables].isna().sum()
```

```
glucose_apache    5386
d1_glucose_max    5386
d1_glucose_min    5386
h1_glucose_max    5386
```

```
h1_glucose_min    5386
dtype: int64
```

```
diabetis['glucose_apache'][diabetis['diabetes_mellitus']==1].isna().sum()
```

```
734
```

```
diabetis['glucose_apache'][diabetis['diabetes_mellitus']==0].isna().sum()
```

```
9621
```

```
X_train6=X_train6.drop('paco2_for_ph_apache', axis=1) #this is not the blood ph we want

selection_for_mean=['bun','albumin', 'creatinine',  'lactate', 'bilirubin',
          'hematocrit', 'hemaglobin', 'sysbp', 'mbp', 'diasbp','sodium', 'potassium', 'hco3']

#glucose already done, single variables not included, differentiate the blood pressure types
#'ph_' will be done apart (below)

for word in selection_for_mean:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_train6[variables].mean(axis=1)
    X_train6[variables]= X_train6[variables].T.fillna(average).T
```

```
Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains('ph_')==True]
```

```
36     paco2_for_ph_apache
38               ph_apache
159     d1_arterial_ph_max
160     d1_arterial_ph_min
167     h1_arterial_ph_max
168     h1_arterial_ph_min
Name: Variable Name, dtype: object
```

```
ph_variables=['ph_apache','d1_arterial_ph_max','d1_arterial_ph_min','h1_arterial_ph_max','h1_arterial_ph_min']
ph_mean=X_train6[ph_variables].mean(axis=1)
X_train6[ph_variables]= X_train6[ph_variables].T.fillna(ph_mean).T
```

```
X_train6.isna().sum() # NaN sum has drecreased indeed
```

```
h1_bun_max                   8980
h1_bun_min                   8980
albumin_apache              49293
d1_albumin_max              49293
d1_albumin_min              49293
h1_albumin_max              49293
h1_albumin_min              49293
creatinine_apache            8649
d1_creatinine_max            8649
d1_creatinine_min            8649
h1_creatinine_max            8649
h1_creatinine_min            8649
d1_lactate_max              66800
d1_lactate_min              66800
h1_lactate_max              66800
h1_lactate_min              66800
bilirubin_apache            52941
d1_bilirubin_max            52941
d1_bilirubin_min            52941
h1_bilirubin_max            52941
h1_bilirubin_min            52941
arf_apache                      0
cirrhosis                       0
bmi                          3158
weight                       2466
age                          3498
gender                         49
                            ...
h1_mbp_invasive_max           241
h1_mbp_invasive_min           241
h1_mbp_max                    241
h1_mbp_min                    241
h1_mbp_noninvasive_max        241
h1_mbp_noninvasive_min        241

h1_sysbp_invasive_max         204
h1_sysbp_invasive_min         204
h1_sysbp_max                  204
```

```
h1_sysbp_min                  204
h1_sysbp_noninvasive_max      204
h1_sysbp_noninvasive_min      204
ph_apache                   59341
d1_arterial_ph_max          59341
d1_arterial_ph_min          59341
h1_arterial_ph_max          59341
h1_arterial_ph_min          59341
sodium_apache                8679
d1_sodium_max                8679
d1_sodium_min                8679
h1_sodium_max                8679
h1_sodium_min                8679
d1_potassium_max             8821
d1_potassium_min             8821
h1_potassium_max             8821
h1_potassium_min             8821
d1_hco3_max                 14074
d1_hco3_min                 14074
h1_hco3_max                 14074
h1_hco3_min                 14074
Length: 100, dtype: int64
```

```
#the same for the test set
X_test6=X_test6.drop('paco2_for_ph_apache', axis=1) #this is not the blood ph we want

selection_for_mean_test=['bun','albumin', 'creatinine',  'lactate', 'bilirubin',
          'hematocrit', 'hemaglobin', 'sysbp', 'mbp', 'diasbp','sodium', 'potassium', 'hco3', 'glucose']

#glucose resinserted, single variables not included, differentiate the blood pressure types
#'ph_' will be done apart (below)

for word in selection_for_mean_test:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_test6[variables].mean(axis=1)
    X_test6[variables]= X_test6[variables].T.fillna(average).T


#the same for _ph
#ph_variables=['ph_apache','d1_arterial_ph_max','d1_arterial_ph_min','h1_arterial_ph_max','h1_arterial_ph_min']
ph_mean=X_test6[ph_variables].mean(axis=1)
```

```
ph_mean=X_test6[ph_variables].mean(axis=1)
X_test6[ph_variables]= X_test6[ph_variables].T.fillna(ph_mean).T


for c in X_train6.columns:
    col_type = X_train6[c].dtype
    if col_type == 'object':
        X_train6[c] = X_train6[c].astype('category')

for c in X_test6.columns:
    col_type = X_test6[c].dtype
    if col_type == 'object':
        X_test6[c] = X_test6[c].astype('category')


import lightgbm as lgb #pip install lightgbm
#kind of optimized model
clf_6 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                                metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                               learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                               n_estimators=5000, max_depth=-1, num_leaves=200 )


#fit the lghbm model
clf_6.fit(X_train6, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                             eval_set= [(X_test6,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featu
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\lightgbm\basic.py:1286: UserWarning: Overriding the parameters from Reference Datase
  warnings.warn('Overriding the parameters from Reference Dataset.')
C:\Users\lucia\Anaconda3\lib\site-packages\lightgbm\basic.py:1098: UserWarning: categorical_column in param dict is overridden.
  warnings.warn('{} in param dict is overridden.'.format(cat_alias))
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.813723
[200]   test's auc: 0.818346
[300]   test's auc: 0.824237
[400]   test's auc: 0.829071
[500]   test's auc: 0.832795
[600]   test's auc: 0.835874
[700]   test's auc: 0.837852
[800]   test's auc: 0.83971
```

```
[900]	test's auc: 0.84078
[1000]	test's auc: 0.841777
[1100]	test's auc: 0.842632
[1200]	test's auc: 0.84321
[1300]	test's auc: 0.843752
[1400]	test's auc: 0.844092
[1500]	test's auc: 0.844371
[1600]	test's auc: 0.844479
[1700]	test's auc: 0.844559
[1800]	test's auc: 0.844762
[1900]	test's auc: 0.844852
[2000]	test's auc: 0.844941
[2100]	test's auc: 0.845063
[2200]	test's auc: 0.845089
[2300]	test's auc: 0.845115
[2400]	test's auc: 0.845104
[2500]	test's auc: 0.845011
Early stopping, best iteration is:
[2350]	test's auc: 0.845169
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=200,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
#it did not improve!!
```

take all variables plus scientific ones with mean imputation

```
flat_list4=['bun_apache',
 'd1_bun_max',
 'd1_bun_min',
 'h1_bun_max',
 'h1_bun_min',
 'albumin_apache',
 'd1_albumin_max',
```

```
'd1_albumin_min',
'h1_albumin_max',
'h1_albumin_min',
'creatinine_apache',
'd1_creatinine_max',
'd1_creatinine_min',
'h1_creatinine_max',
'h1_creatinine_min',
'd1_lactate_max',
'd1_lactate_min',
'h1_lactate_max',
'h1_lactate_min',
'bilirubin_apache',
'd1_bilirubin_max',
'd1_bilirubin_min',
'h1_bilirubin_max',
'h1_bilirubin_min',
'arf_apache',
'cirrhosis',
'bmi',
'weight',
'age',
'gender',
'ethnicity',
'glucose_apache',
'd1_glucose_max',
'd1_glucose_min',
'h1_glucose_max',
'h1_glucose_min',
'urineoutput_apache',
'hematocrit_apache',
'd1_hematocrit_max',
'd1_hematocrit_min',
'h1_hematocrit_max',
'h1_hematocrit_min',
'd1_hemaglobin_max',
'd1_hemaglobin_min',
'h1_hemaglobin_max',
```

```
'h1_hemaglobin_min',
'd1_diasbp_invasive_max',
'd1_diasbp_invasive_min',
'd1_diasbp_max',
'd1_diasbp_min',
'd1_diasbp_noninvasive_max',
'd1_diasbp_noninvasive_min',
'd1_mbp_invasive_max',
'd1_mbp_invasive_min',
'd1_mbp_max',
'd1_mbp_min',
'd1_mbp_noninvasive_max',
'd1_mbp_noninvasive_min',
'd1_sysbp_invasive_max',
'd1_sysbp_invasive_min',
'd1_sysbp_max',
'd1_sysbp_min',
'd1_sysbp_noninvasive_max',
'd1_sysbp_noninvasive_min',
'h1_diasbp_invasive_max',
'h1_diasbp_invasive_min',
'h1_diasbp_max',
'h1_diasbp_min',
'h1_diasbp_noninvasive_max',
'h1_diasbp_noninvasive_min',
'h1_mbp_invasive_max',
'h1_mbp_invasive_min',
'h1_mbp_max',
'h1_mbp_min',
'h1_mbp_noninvasive_max',
'h1_mbp_noninvasive_min',
'h1_sysbp_invasive_max',
'h1_sysbp_invasive_min',
'h1_sysbp_max',
'h1_sysbp_min',
'h1_sysbp_noninvasive_max',
'h1_sysbp_noninvasive_min',
'ph_apache',
```

```
 'ph_apache',
 'd1_arterial_ph_max',
 'd1_arterial_ph_min',
 'h1_arterial_ph_max',
 'h1_arterial_ph_min',
 'sodium_apache',
 'd1_sodium_max',
 'd1_sodium_min',
 'h1_sodium_max',
 'h1_sodium_min',
 'd1_potassium_max',
 'd1_potassium_min',
 'h1_potassium_max',
 'h1_potassium_min',
 'd1_hco3_max',
 'd1_hco3_min',
 'h1_hco3_max',
 'h1_hco3_min']
```

```
X_train[flat_list4]=X_train6
X_test[flat_list4]=X_test6
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-eed66b9d5967> in <module>()
----> 1 X_train[flat_list4]=X_train6
      2 X_test[flat_list4]=X_test6

NameError: name 'X_train6' is not defined
```

SEARCH STACK OVERFLOW

```
for c in X_test.columns:
    col_type = X_test[c].dtype
    if col_type == 'object':
        X_test[c] = X_test[c].astype('category')

for c in X_train.columns:
```

```
for c in X_train.columns:
    col_type = X_train[c].dtype
    if col_type == 'object':
        X_train[c] = X_train[c].astype('category')
```

```
clf_6 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                                metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                               learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                               n_estimators=5000, max_depth=-1, num_leaves=100)
```

```
#fit the lghbm model
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                             eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

what about imputing the mean in all the available variables?

```
#difers=set(Data_dictionary['Variable Name'])-set(flat_list4)
#difers
list_difers=['pco2', 'po2', 'calcium', 'heartrate', 'inr', 'pao2fio2ratio', 'platelets', 'resprate', 'spo2', 'temp', 'wbc']
```

```
for word in list_difers:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_train[variables].mean(axis=1)
    X_train[variables]= X_train[variables].T.fillna(average).T
```

```
clf_6 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                                metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                               learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                               n_estimators=5000, max_depth=-1, num_leaves=50)
```

```
#fit the lghbm model
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
```

```
                              eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur

Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.820639
[200]   test's auc: 0.829509
[300]   test's auc: 0.83536
[400]   test's auc: 0.840432
[500]   test's auc: 0.844014
[600]   test's auc: 0.847101
[700]   test's auc: 0.849289
[800]   test's auc: 0.851071
[900]   test's auc: 0.852283
[1000]  test's auc: 0.853292
[1100]  test's auc: 0.854207
[1200]  test's auc: 0.854949
[1300]  test's auc: 0.855583
[1400]  test's auc: 0.856124
[1500]  test's auc: 0.85658
[1600]  test's auc: 0.856973
[1700]  test's auc: 0.857297
[1800]  test's auc: 0.857585
[1900]  test's auc: 0.857826
[2000]  test's auc: 0.858017
[2100]  test's auc: 0.858174
[2200]  test's auc: 0.858312
[2300]  test's auc: 0.858451
[2400]  test's auc: 0.85856
[2500]  test's auc: 0.858694
[2600]  test's auc: 0.858789
[2700]  test's auc: 0.858911
[2800]  test's auc: 0.85899
[2900]  test's auc: 0.859025
[3000]  test's auc: 0.859056
[3100]  test's auc: 0.859136
[3200]  test's auc: 0.859147
[3300]  test's auc: 0.8592
[3400]  test's auc: 0.859236
[3500]  test's auc: 0.859273
[3600]  test's auc: 0.859271
[3700]  test's auc: 0.859295
[3800]  test's auc: 0.85932
```

```
[3900]	test's auc: 0.859305
[4000]	test's auc: 0.859361
[4100]	test's auc: 0.859397
[4200]	test's auc: 0.859418
[4300]	test's auc: 0.859461
[4400]	test's auc: 0.859468
[4500]	test's auc: 0.859451
[4600]	test's auc: 0.859456
Early stopping, best iteration is:
[4378]	test's auc: 0.859492
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=50,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

what about use only the mean in all the similar variables?

```
#list_difers=['pco2', 'po2', 'calcium', 'heartrate', 'inr', 'pao2fio2ratio', 'platelets', 'resprate', 'spo2', 'temp', 'wbc']

lista4=[] # list of the relevant features
for word in list_difers:
    a=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    lista4.append(a.to_list())

# converts to single list
flat_list5 = [item for sublista in lista4 for item in sublista]
```

```
#other_difference=set(Data_dictionary['Variable Name'])-set(flat_list4)-set(flat_list5)
other_difference_=['aids',
 'apache_2_diagnosis',
 'apache_3j_diagnosis',
 'apache_post_operative',
```

```
 'elective_surgery',
 'fio2_apache',
 'gcs_eyes_apache',
 'gcs_motor_apache',
 'gcs_unable_apache',
 'gcs_verbal_apache',
 'heart_rate_apache',
 'height',
 'hepatic_failure',
 'hospital_admit_source',
 'icu_admit_source',
 'icu_stay_type',
 'icu_type',
 'immunosuppression',
 'intubated_apache',
 'leukemia',
 'lymphoma',
 'map_apache',
 'paco2_apache',
 'paco2_for_ph_apache',
 'pao2_apache',
 'pre_icu_los_days',
 'readmission_status',
 'solid_tumor_with_metastasis',
 'ventilated_apache']


X_train_=X_train[other_difference_]


X_test_=X_test[other_difference_]


selection_main=['bun','albumin', 'creatinine',  'lactate', 'bilirubin','hematocrit', 'hemaglobin', 'sysbp',
                    'mbp', 'diasbp','sodium', 'potassium', 'hco3', 'glucose', 'pco2', 'po2', 'calcium',
                    'heartrate', 'inr', 'pao2fio2ratio', 'platelets', 'resprate', 'spo2', 'temp', 'wbc']

#'ph_' will be done apart (below)
```

```
for word in selection_main:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_train[variables].mean(axis=1)
    X_train_[word]= average

#the same for _ph
ph_variables=['ph_apache','d1_arterial_ph_max','d1_arterial_ph_min','h1_arterial_ph_max','h1_arterial_ph_min']
ph_mean=X_train[ph_variables].mean(axis=1)
X_train_['ph_']= ph_mean
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  # Remove the CWD from sys.path while we load stuff.
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  from ipykernel import kernelapp as app
```

```
#The same for testset
for word in selection_main:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_test[variables].mean(axis=1)
    X_test_[word]= average

#the same for _ph
ph_mean=X_test[ph_variables].mean(axis=1)
X_test_['ph_']= ph_mean
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
  """
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

```
X_train_
```

	aids	apache_2_diagnosis	apache_3j_diagnosis	apache_post_operative	elective_surgery	fio2_apache	gcs_eyes_apache	g
91852	0	113.0	501.06	0	0	NaN	4.0	
44993	0	123.0	702.01	0	0	NaN	4.0	
129445	0	113.0	501.05	0	0	0.75	2.0	
55299	0	301.0	1506.07	1	0	0.40	1.0	
102655	0	112.0	107.01	0	0	NaN	4.0	
101835	0	120.0	407.01	0	0	1.00	2.0	
70820	0	NaN	0.25	0	1	NaN	4.0	
124784	0	213.0	1405.03	1	1	NaN	4.0	
4630	0	301.0	403.01	0	0	NaN	4.0	
27642	0	308.0	1903.03	1	1	NaN	4.0	
23732	0	202.0	1204.01	1	1	NaN	3.0	
58707	0	217.0	1502.02	1	1	1.00	1.0	
114763	0	213.0	1405.07	1	1	NaN	3.0	
95966	0	213.0	1405.02	1	1	NaN	4.0	
103129	0	301.0	403.01	0	0	0.30	2.0	
16573	0	117.0	106.01	0	0	NaN	4.0	
19466	0	202.0	1211.01	1	1	NaN	3.0	
43460	0	113.0	501.05	0	0	NaN	4.0	
93035	0	207.0	1602.12	1	1	NaN	3.0	
76849	0	119.0	601.01	0	0	NaN	4.0	
79013	0	301.0	403.01	0	0	NaN	3.0	
33344	0	303.0	211.10	0	0	NaN	1.0	

99504	0	302.0	111.01	0	0	NaN	4.0
56798	0	113.0	502.01	0	0	NaN	4.0
897	0	124.0	306.01	0	0	NaN	4.0
23356	0	102.0	206.01	0	0	NaN	3.0
24257	0	110.0	104.01	0	0	NaN	4.0
128883	0	112.0	107.01	0	0	0.40	4.0
126796	0	112.0	107.01	0	0	NaN	4.0
42657	0	113.0	501.04	0	0	NaN	3.0
...	...	...	...	...	...	...	...
103355	0	117.0	106.01	0	0	NaN	4.0
5311	0	301.0	1506.09	1	1	NaN	4.0
67969	0	113.0	501.05	0	0	1.00	2.0
121637	0	119.0	601.03	0	0	0.80	4.0
64925	0	302.0	109.07	0	0	NaN	4.0
62955	0	124.0	305.01	0	0	NaN	4.0
59735	0	113.0	501.05	0	0	0.25	4.0
769	0	304.0	311.01	0	0	1.00	4.0
64820	0	303.0	211.12	0	0	NaN	4.0
67221	0	122.0	703.03	0	0	NaN	4.0
41090	0	113.0	501.06	0	0	NaN	4.0
16023	0	213.0	1405.02	1	1	NaN	4.0

```
for c in X_test_.columns:
    col_type = X_test_[c].dtype
```

```
    if col_type == 'object':
        X_test_[c] = X_test_[c].astype('category')

for c in X_train_.columns:
    col_type = X_train_[c].dtype
    if col_type == 'object':
        X_train_[c] = X_train_[c].astype('category')
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
C:\Users\lucia\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

```
#fit the lghbm model
clf_6.fit(X_train_, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test_,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featu
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.787981
[200]   test's auc: 0.789993
[300]   test's auc: 0.79199
[400]   test's auc: 0.797154
[500]   test's auc: 0.799644
[600]   test's auc: 0.802104
[700]   test's auc: 0.804391
[800]   test's auc: 0.80562
[900]   test's auc: 0.806989
[1000]  test's auc: 0.808019
[1100]  test's auc: 0.808707
[1200]  test's auc: 0.809369
[1300]  test's auc: 0.809916
```

```
[1400]	test's auc: 0.810303
[1500]	test's auc: 0.810633
[1600]	test's auc: 0.810955
[1700]	test's auc: 0.811254
[1800]	test's auc: 0.811498
[1900]	test's auc: 0.811647
[2000]	test's auc: 0.811807
[2100]	test's auc: 0.81196
[2200]	test's auc: 0.812073
[2300]	test's auc: 0.812129
[2400]	test's auc: 0.812195
[2500]	test's auc: 0.812269
[2600]	test's auc: 0.812317
[2700]	test's auc: 0.812374
[2800]	test's auc: 0.812412
[2900]	test's auc: 0.812476
[3000]	test's auc: 0.812549
[3100]	test's auc: 0.812617
[3200]	test's auc: 0.812659
[3300]	test's auc: 0.812695
[3400]	test's auc: 0.812718
[3500]	test's auc: 0.812692
[3600]	test's auc: 0.812683
Early stopping, best iteration is:
[3424]	test's auc: 0.812742
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=50,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

TAKE ALL VARIABLES SO FAR ALWAYS BETTER THAN PRE-SELECTING VARIABLES!!

```
# test all varibales except glucose ones
glucose_variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains('glucose')==True]
Xtrain_WithoutGlucose=X_train.drop(glucose_variables, axis=1)
Xtest_WithoutGlucose=X_test.drop(glucose_variables, axis=1)
```

```
#transform object variables into categgory type for LGBM
for c in Xtrain_WithoutGlucose.columns:
    col_type = Xtrain_WithoutGlucose[c].dtype
    if col_type == 'object':
        Xtrain_WithoutGlucose[c] = Xtrain_WithoutGlucose[c].astype('category')


#transform object variables into categgory type for LGBM
for c in Xtest_WithoutGlucose.columns:
    col_type = Xtest_WithoutGlucose[c].dtype
    if col_type == 'object':
        Xtest_WithoutGlucose[c] = Xtest_WithoutGlucose[c].astype('category')


#import lightgbm as lgb

clf_6 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                               metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                              learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                              n_estimators=5000, max_depth=-1, num_leaves=50)


#fit the lghbm model
clf_6.fit(Xtrain_WithoutGlucose, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(Xtest_WithoutGlucose,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', cate
```

```
C:\Users\lucia\Anaconda3\lib\site-packages\lightgbm\basic.py:1286: UserWarning: Overriding the parameters from Reference Datase
  warnings.warn('Overriding the parameters from Reference Dataset.')
C:\Users\lucia\Anaconda3\lib\site-packages\lightgbm\basic.py:1098: UserWarning: categorical_column in param dict is overridden.
  warnings.warn('{} in param dict is overridden.'.format(cat_alias))
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.724811
[200]   test's auc: 0.73775
[300]   test's auc: 0.747027
[400]   test's auc: 0.752943
[500]   test's auc: 0.757085
[600]   test's auc: 0.759872
[700]   test's auc: 0.762277
[800]   test's auc: 0.763875
[900]   test's auc: 0.765261
```

```
[1000]	test's auc: 0.766258
[1100]	test's auc: 0.767183
[1200]	test's auc: 0.767731
[1300]	test's auc: 0.7684
[1400]	test's auc: 0.768877
[1500]	test's auc: 0.769241
[1600]	test's auc: 0.769488
[1700]	test's auc: 0.769699
[1800]	test's auc: 0.769848
[1900]	test's auc: 0.770015
[2000]	test's auc: 0.770119
[2100]	test's auc: 0.770264
[2200]	test's auc: 0.77041
[2300]	test's auc: 0.770534
[2400]	test's auc: 0.770529
[2500]	test's auc: 0.770631
[2600]	test's auc: 0.770674
[2700]	test's auc: 0.770733
[2800]	test's auc: 0.770694
[2900]	test's auc: 0.770645
[3000]	test's auc: 0.770673
Early stopping, best iteration is:
[2782]	test's auc: 0.770743
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=50,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
#there is no way, glucose analysis is always needed for better fitting!
```

what about adding the mean calculated features? i.e. 180 varibales + 26?

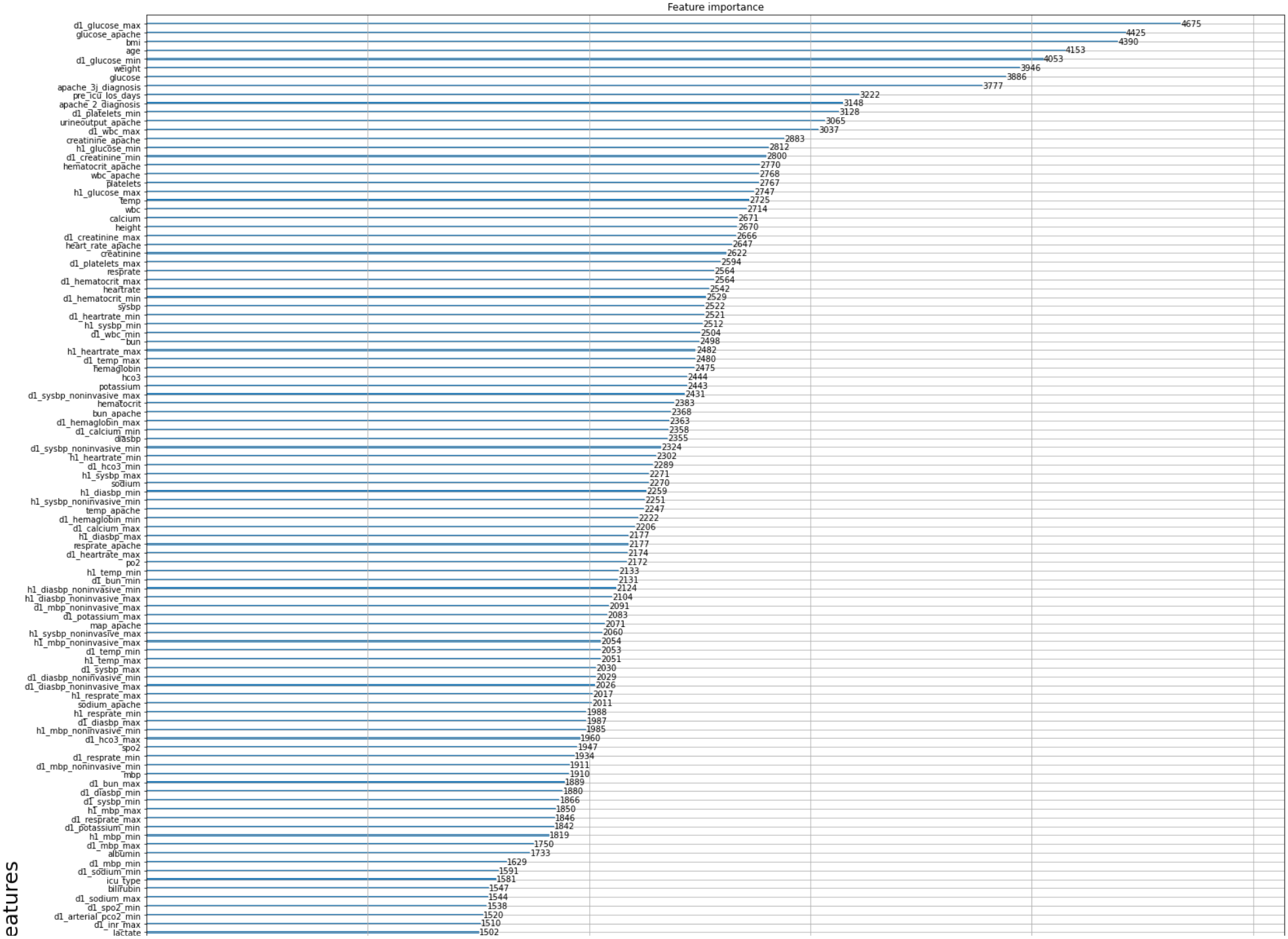
```
selection_main=['bun','albumin', 'creatinine',  'lactate', 'bilirubin','hematocrit', 'hemaglobin', 'sysbp',
                    'mbp', 'diasbp','sodium', 'potassium', 'hco3', 'glucose', 'pco2', 'po2', 'calcium',
                    'heartrate', 'inr', 'pao2fio2ratio', 'platelets', 'resprate', 'spo2', 'temp', 'wbc'] #25 extra features
```

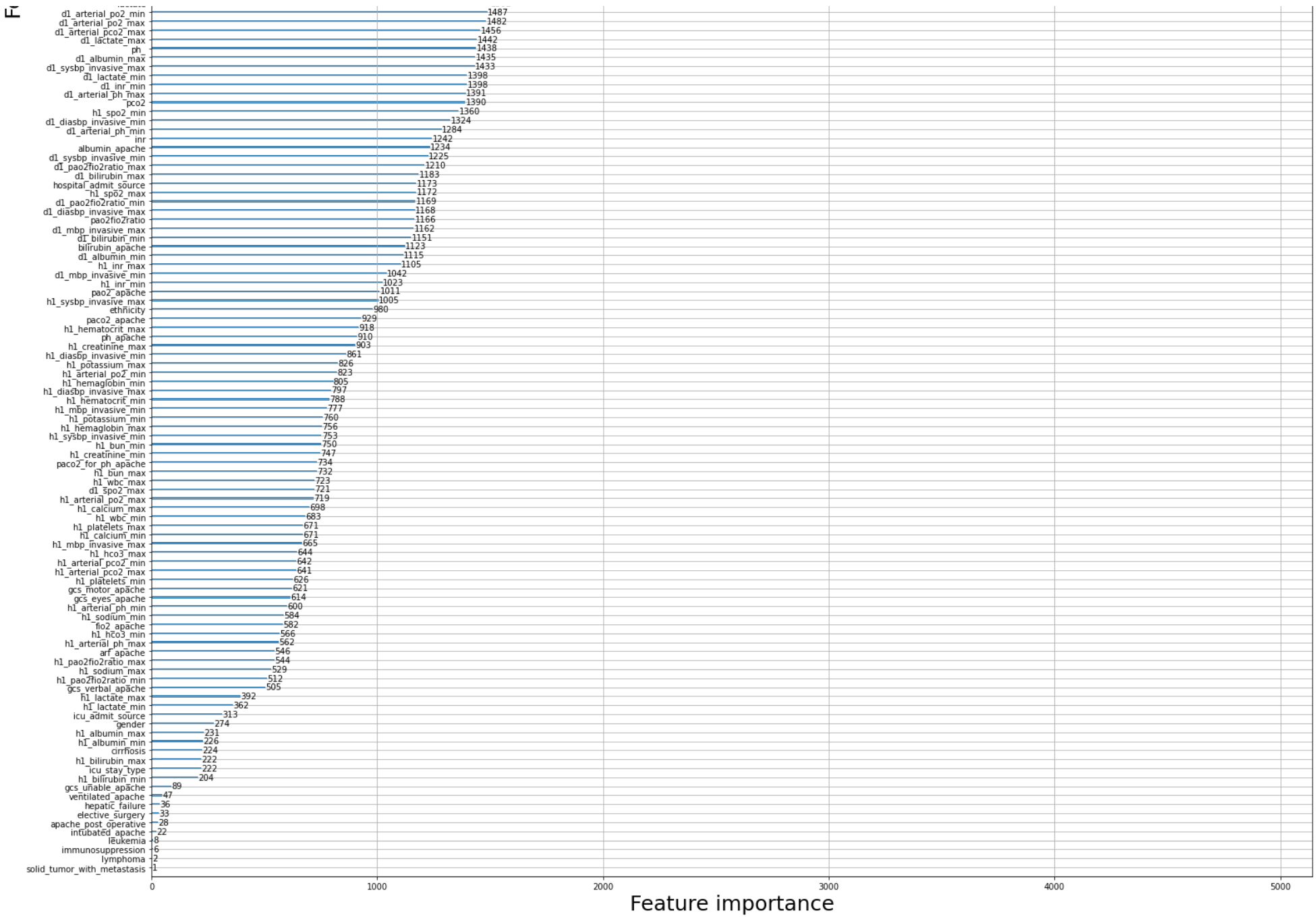
```
#'ph_' will be done apart (below)

for word in selection_main:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_train[variables].mean(axis=1)
    X_train[word]= average

#the same for _ph
ph_variables=['ph_apache','d1_arterial_ph_max','d1_arterial_ph_min','h1_arterial_ph_max','h1_arterial_ph_min']
ph_mean=X_train[ph_variables].mean(axis=1)
X_train['ph_']= ph_mean


#The same for testset
for word in selection_main:
    variables=Data_dictionary['Variable Name'][Data_dictionary['Variable Name'].str.contains(word)==True]
    average=X_test[variables].mean(axis=1)
    X_test[word]= average

#the same for _ph
ph_mean=X_test[ph_variables].mean(axis=1)
X_test['ph_']= ph_mean


import lightgbm as lgb

clf_6 = lgb.LGBMClassifier(boosting_type='gbdt', objective='binary',n_jobs=-1,
                              metric= 'auc', scale_pos_weight= 0.5, subsample = 1, force_col_wise = True,
                             learning_rate= 0.01, colsample_bytree= 0.2, reg_alpha= 3, reg_lambda= 1,
                             n_estimators=5000, max_depth=-1, num_leaves=100)


#fit the lghbm model
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.825104
```

```
[200]	test's auc: 0.832359
[300]	test's auc: 0.838963
[400]	test's auc: 0.843508
[500]	test's auc: 0.84702
[600]	test's auc: 0.849774
[700]	test's auc: 0.851811
[800]	test's auc: 0.853614
[900]	test's auc: 0.854834
[1000]	test's auc: 0.855836
[1100]	test's auc: 0.856687
[1200]	test's auc: 0.857335
[1300]	test's auc: 0.857805
[1400]	test's auc: 0.858274
[1500]	test's auc: 0.858648
[1600]	test's auc: 0.858931
[1700]	test's auc: 0.859276
[1800]	test's auc: 0.859532
[1900]	test's auc: 0.859718
[2000]	test's auc: 0.859905
[2100]	test's auc: 0.860039
[2200]	test's auc: 0.86012
[2300]	test's auc: 0.860229
[2400]	test's auc: 0.860309
[2500]	test's auc: 0.860425
[2600]	test's auc: 0.860459
[2700]	test's auc: 0.86053
[2800]	test's auc: 0.860601
[2900]	test's auc: 0.860648
[3000]	test's auc: 0.860661
[3100]	test's auc: 0.860684
[3200]	test's auc: 0.860703
[3300]	test's auc: 0.860733
[3400]	test's auc: 0.860679
[3500]	test's auc: 0.860701
Early stopping, best iteration is:
[3276]	test's auc: 0.860734
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=100,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
import matplotlib.pyplot as plt
plt.rcParams['axes.labelsize'] = 25

lgb.plot_importance(clf_6, figsize = (25,40))
plt.show()
```


Feature importance
d1_glucose_max 4675
glucose_apache 4425
bmi 4390
age 4153
d1_glucose_min 4053
weight 3946
glucose 3886
apache_3j_diagnosis 3777
pre_icu_los_days 3222
apache_2_diagnosis 3148
d1_platelets_min 3128
urineoutput_apache 3065
d1_wbc_max 3037
creatinine_apache 2883
h1_glucose_min 2812
d1_creatinine_min 2800
hematocrit_apache 2770
wbc_apache 2768
platelets 2767
h1_glucose_max 2747
temp 2725
wbc 2714
calcium 2671
height 2670
d1_creatinine_max 2666
heart_rate_apache 2647
creatinine 2622
d1_platelets_max 2594
resprate 2564
d1_hematocrit_max 2564
heartrate 2542
d1_hematocrit_min 2529
sysbp 2522
d1_heartrate_min 2521
h1_sysbp_min 2512
d1_wbc_min 2504
bun 2498
h1_heartrate_max 2482
d1_temp_max 2480
hemaglobin 2475
hco3 2444
potassium 2443
d1_sysbp_noninvasive_max 2431
hematocrit 2383
bun_apache 2368
d1_hemaglobin_max 2363
d1_calcium_min 2358
diasbp 2355
d1_sysbp_noninvasive_min 2324
h1_heartrate_min 2302
d1_hco3_min 2289
h1_sysbp_max 2271
sodium 2270
h1_diasbp_min 2259
h1_sysbp_noninvasive_min 2251
temp_apache 2247
d1_hemaglobin_min 2222
d1_calcium_max 2206
h1_diasbp_max 2177
resprate_apache 2177
d1_heartrate_max 2174
po2 2172
h1_temp_min 2133
d1_bun_min 2131
h1_diasbp_noninvasive_min 2124
h1_diasbp_noninvasive_max 2104
d1_mbp_noninvasive_max 2091
d1_potassium_max 2083
map_apache 2071
h1_sysbp_noninvasive_max 2060
h1_mbp_noninvasive_max 2054
d1_temp_min 2053
h1_temp_max 2051
d1_sysbp_max 2030
d1_diasbp_noninvasive_min 2029
d1_diasbp_noninvasive_max 2026
h1_resprate_max 2017
sodium_apache 2011
h1_resprate_min 1988
d1_diasbp_max 1987
h1_mbp_noninvasive_min 1985
d1_hco3_max 1960
spo2 1947
d1_resprate_min 1934
d1_mbp_noninvasive_min 1911
mbp 1910
d1_bun_max 1889
d1_diasbp_min 1880
d1_sysbp_min 1866
h1_mbp_max 1850
d1_resprate_max 1846
d1_potassium_min 1842
h1_mbp_min 1819
d1_mbp_max 1750
albumin 1733
d1_mbp_min 1629
d1_sodium_min 1591
icu_type 1581
bilirubin 1547
d1_sodium_max 1544
d1_spo2_min 1538
d1_arterial_pco2_min 1520
d1_inr_max 1510
lactate 1502
eatures




```
X_train[['bmi', 'weight', 'height']].isna().sum()
```

```
bmi       3158
weight    2466
height    1431
dtype: int64
```

```
X_train['age'].isna().sum()
```

```
3498
```

```
X_train[['weight', 'height']].describe()
```

	weight	height
count	88643.000000	89678.000000
mean	83.791841	169.622144
std	24.954628	10.828152
min	38.600000	137.200000
25%	66.600000	162.500000
50%	80.000000	170.100000
75%	96.840000	177.800000
max	186.000000	195.590000

```
X_train[X_train.gender=='M'][['weight', 'height', 'age']].describe()
#X_train.gender
```

	weight	height	age
count	49324.000000	49324.000000	49324.000000
mean	89.187808	176.235703	61.563134
std	23.618900	8.311854	16.192904
min	38.600000	137.200000	0.000000

```
X_train[X_train.gender=='F'][['weight', 'height', 'age']].describe()
```

	weight	height	age
count	41736.000000	41736.000000	41736.000000
mean	77.192701	161.804945	62.559397
std	24.205653	7.578239	16.882042
min	38.600000	137.200000	0.000000
25%	60.000000	157.400000	53.000000
50%	73.000000	162.600000	64.000000
75%	88.500000	167.600000	76.000000
max	186.000000	195.590000	89.000000

```
#use the mean or median height, weight, age when there is no information
X_train['height'] = X_train['height'].fillna(X_train.groupby("gender")["height"].transform("mean"))
X_train['weight'] = X_train['weight'].fillna(X_train.groupby("gender")["weight"].transform("median"))
X_train['age'] = X_train['age'].fillna(X_train.groupby("gender")["age"].transform("median"))


#the same for test
X_test['height'] = X_test['height'].fillna(X_test.groupby("gender")["height"].transform("mean"))
X_test['weight'] = X_test['weight'].fillna(X_test.groupby("gender")["weight"].transform("median"))
X_test['age'] = X_test['age'].fillna(X_test.groupby("gender")["age"].transform("median"))
```

```
X_train.isna().sum()
```

```
age                              13
bmi                            3158
elective_surgery                  0
ethnicity                      1112
gender                           49
height                           16
hospital_admit_source         23276
icu_admit_source                157
icu_stay_type                     0
icu_type                          0
pre_icu_los_days                  0
weight                           15
albumin_apache                54832
apache_2_diagnosis             1185
apache_3j_diagnosis             609
apache_post_operative             0
arf_apache                        0
bilirubin_apache              57943
bun_apache                    17912
creatinine_apache             17535
fio2_apache                   69828
gcs_eyes_apache                1535
gcs_motor_apache               1535
gcs_unable_apache               503
gcs_verbal_apache              1535
glucose_apache                10355
heart_rate_apache               219
hematocrit_apache             18876
intubated_apache                  0
map_apache                      303
                              ...
h1_potassium_max              70621
h1_potassium_min              70621
h1_sodium_max                 71292
h1_sodium_min                 71292
h1_wbc_max                    74242
h1_wbc_min                    74242
d1_arterial_pco2_max          59180
d1_arterial_pco2_min          59180
```

```
d1_arterial_ph_max            59475
d1_arterial_ph_min            59475
d1_arterial_po2_max           58877
d1_arterial_po2_min           58877
d1_pao2fio2ratio_max          65358
d1_pao2fio2ratio_min          65358
h1_arterial_pco2_max          75379
h1_arterial_pco2_min          75379
h1_arterial_ph_max            75539
h1_arterial_ph_min            75539
h1_arterial_po2_max           75237
h1_arterial_po2_min           75237
h1_pao2fio2ratio_max          79378
h1_pao2fio2ratio_min          79378
aids                              0
cirrhosis                         0
hepatic_failure                   0
immunosuppression                 0
leukemia                          0
lymphoma                          0
```

```
#bmi seems important
#calculate bmi from weight and height, formula= weight/(height**2)
X_train['bmi']=X_train['bmi'].fillna(X_train['weight']/((X_train['height']/100)**2))
X_test['bmi']=X_test['bmi'].fillna(X_test['weight']/((X_test['height']/100)**2))
```

```
X_train['bmi'].isna().sum()
```

```
18
```

```
#do the same for test
X_test[['weight', 'height']].describe() #nearly same values as train
```

	weight	height
count	38051.000000	38402.000000
mean	83.789387	169.572365
std	24.983030	10.844658
min	38.600000	137.200000
25%	66.500000	162.500000

```
X_train[['bmi']].describe()
```

	bmi	bmi_cal
count	87951.000000	91091.000000
mean	29.103131	29.018061
std	8.262463	8.048844
min	14.844926	10.362694
25%	23.582766	23.673925
50%	27.555611	27.560116
75%	32.812500	32.607268
max	67.814990	98.810870

```
#fit again
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.824545
[200]   test's auc: 0.83171
[300]   test's auc: 0.83836
[400]   test's auc: 0.843024
```

```
[500]   test's auc: 0.846629
[600]   test's auc: 0.849434
[700]   test's auc: 0.851562
[800]   test's auc: 0.853358
[900]   test's auc: 0.854609
[1000]  test's auc: 0.855692
[1100]  test's auc: 0.856552
[1200]  test's auc: 0.857181
[1300]  test's auc: 0.857592
[1400]  test's auc: 0.857986
[1500]  test's auc: 0.858362
[1600]  test's auc: 0.858689
[1700]  test's auc: 0.858984
[1800]  test's auc: 0.8592
[1900]  test's auc: 0.859368
[2000]  test's auc: 0.859503
[2100]  test's auc: 0.859665
[2200]  test's auc: 0.859817
[2300]  test's auc: 0.859943
[2400]  test's auc: 0.859994
[2500]  test's auc: 0.860067
[2600]  test's auc: 0.860137
[2700]  test's auc: 0.860207
[2800]  test's auc: 0.860221
[2900]  test's auc: 0.86023
[3000]  test's auc: 0.860248
[3100]  test's auc: 0.860247
[3200]  test's auc: 0.860295
[3300]  test's auc: 0.860341
[3400]  test's auc: 0.860326
[3500]  test's auc: 0.86032
Early stopping, best iteration is:
[3303]  test's auc: 0.860346
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=100,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
clf_10 = lgb.LGBMClassifier(boosting_type='gbdt', objective='cross_entropy', n_jobs=-1,
                            metric= 'auc', scale_pos_weight= 5, subsample = 1, force_col_wise = True,
                            learning_rate= 0.005, colsample_bytree= 0.8, reg_alpha= 3, reg_lambda= 1,
                            n_estimators= 50000, max_depth= 1, num_leaves= 6)
```

```
                              n_estimators = 50000, max_depth=-1, num_leaves=6)
```

```
len(X_train.columns)
```

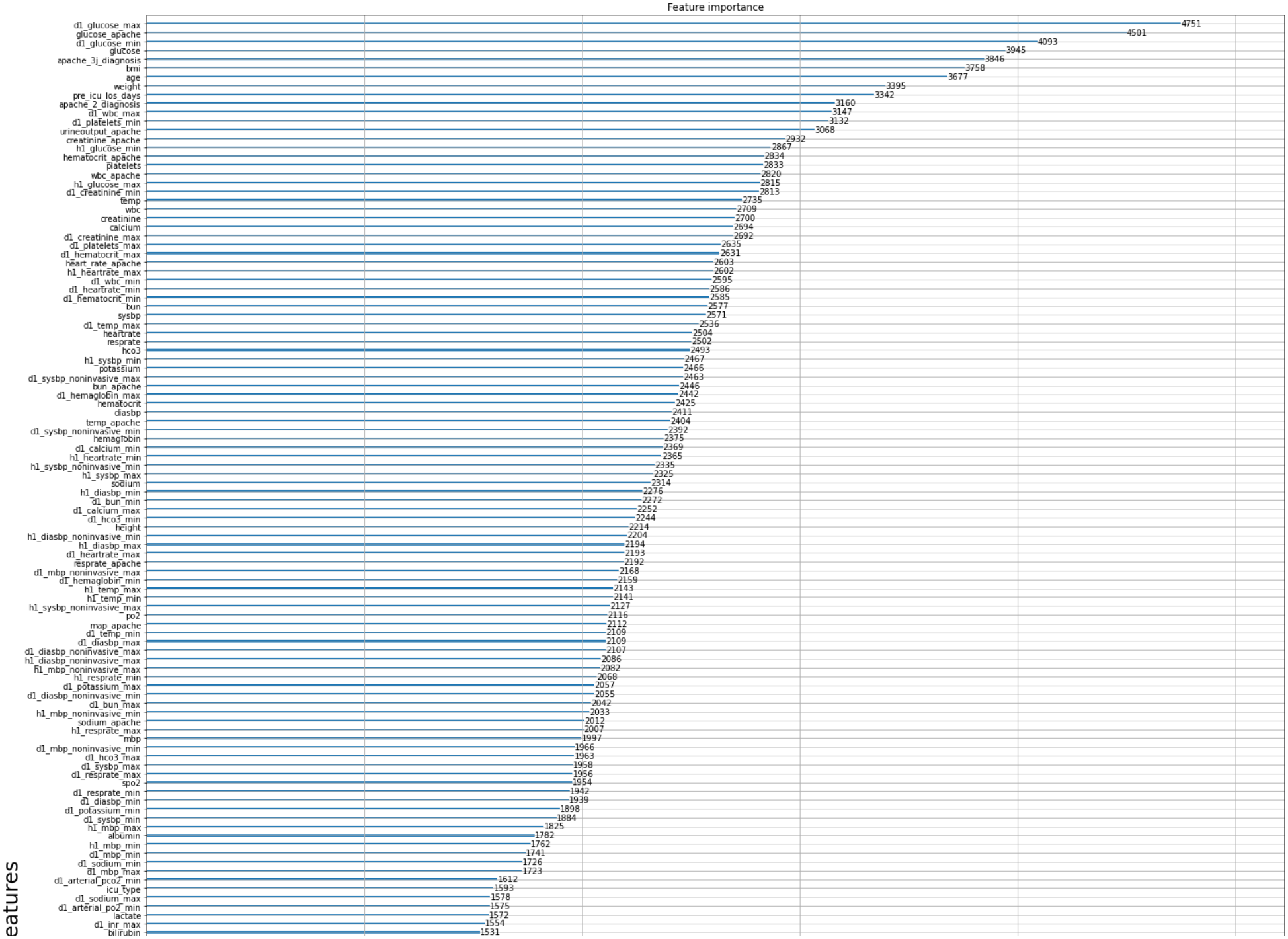
```
201
```

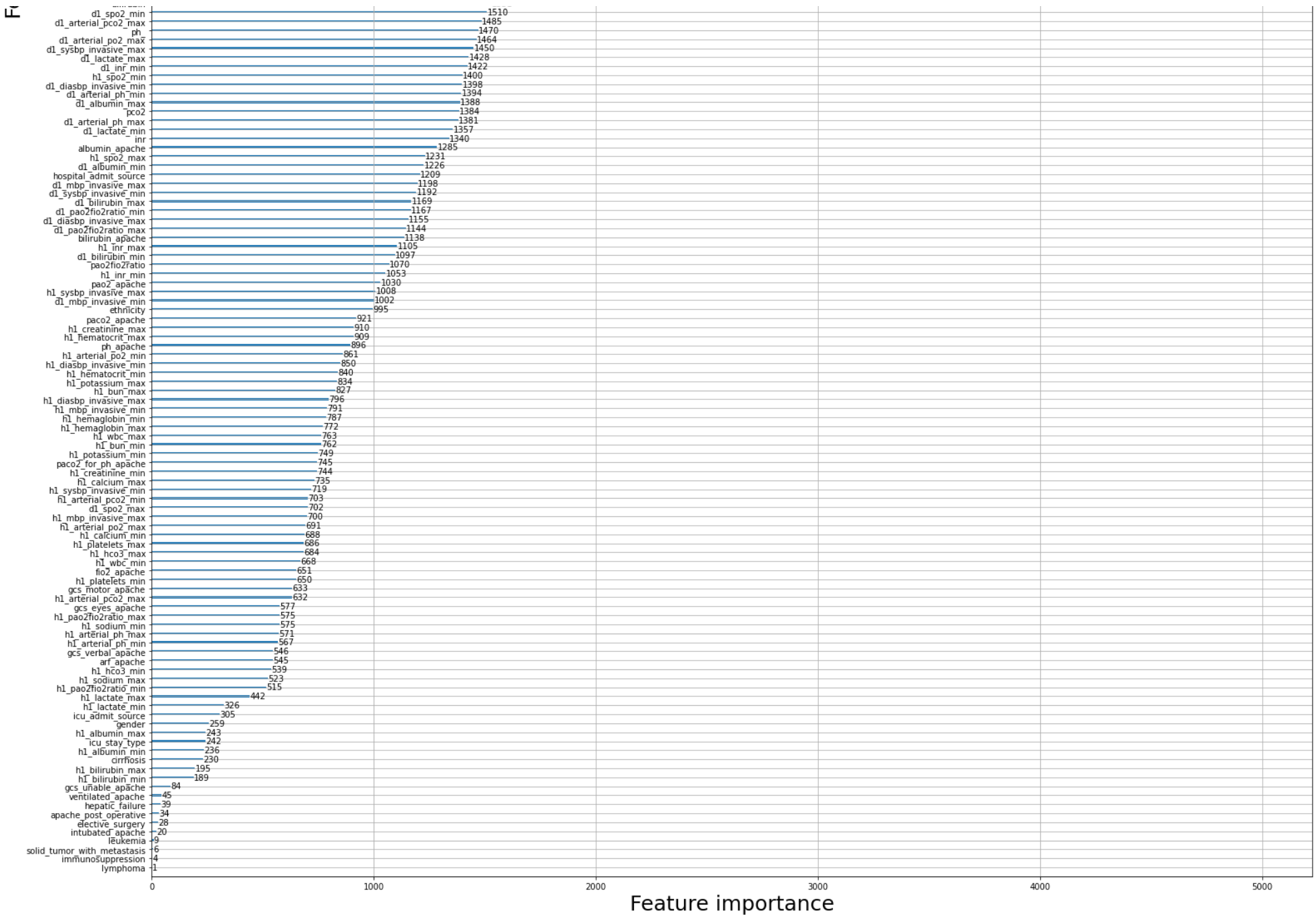
```
#fit again
clf_10.fit(X_train, y_train, early_stopping_rounds= 300, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 300 rounds
[100]   test's auc: 0.80031
[200]   test's auc: 0.806461
[300]   test's auc: 0.811241
[400]   test's auc: 0.815466
[500]   test's auc: 0.81948
[600]   test's auc: 0.822974
[700]   test's auc: 0.825818
[800]   test's auc: 0.828309
[900]   test's auc: 0.830393
[1000]  test's auc: 0.832205
[1100]  test's auc: 0.833772
[1200]  test's auc: 0.835184
[1300]  test's auc: 0.836483
[1400]  test's auc: 0.837653
[1500]  test's auc: 0.838703
[1600]  test's auc: 0.839699
[1700]  test's auc: 0.840598
[1800]  test's auc: 0.841413
[1900]  test's auc: 0.842204
[2000]  test's auc: 0.842864
[2100]  test's auc: 0.8435
[2200]  test's auc: 0.844063
[2300]  test's auc: 0.844606
[2400]  test's auc: 0.845103
[2500]  test's auc: 0.845549
[2600]  test's auc: 0.845976
[2700]  test's auc: 0.846365
[2800]  test's auc: 0.846741
[2900]  test's auc: 0.847089
```

```
[3000]	test's auc: 0.847436
[3100]	test's auc: 0.847788
[3200]	test's auc: 0.848122
[3300]	test's auc: 0.848423
[3400]	test's auc: 0.848721
[3500]	test's auc: 0.848956
[3600]	test's auc: 0.849185
[3700]	test's auc: 0.849422
[3800]	test's auc: 0.849618
[3900]	test's auc: 0.849826
[4000]	test's auc: 0.849988
[4100]	test's auc: 0.850161
[4200]	test's auc: 0.850312
[4300]	test's auc: 0.850459
[4400]	test's auc: 0.850599
[4500]	test's auc: 0.850734
[4600]	test's auc: 0.850872
[4700]	test's auc: 0.850996
[4800]	test's auc: 0.851113
[4900]	test's auc: 0.851229
[5000]	test's auc: 0.851335
[5100]	test's auc: 0.851462
[5200]	test's auc: 0.851568
[5300]	test's auc: 0.851682
[5400]	test's auc: 0.851774
[5500]	test's auc: 0.851872
[5600]	test's auc: 0.851955
[5700]	test's auc: 0.85205
[5800]	test's auc: 0.852156
```

```
import matplotlib.pyplot as plt
plt.rcParams['axes.labelsize'] = 25

lgb.plot_importance(clf_6, figsize = (25,40))
plt.show()
```


Feature importance
d1_glucose_max 4751
glucose_apache 4501
d1_glucose_min 4093
glucose 3945
apache_3j_diagnosis 3846
bmi 3758
age 3677
weight 3395
pre_icu_los_days 3342
apache_2_diagnosis 3160
d1_wbc_max 3147
d1_platelets_min 3132
urineoutput_apache 3068
creatinine_apache 2932
h1_glucose_min 2867
hematocrit_apache 2834
platelets 2833
wbc_apache 2820
h1_glucose_max 2815
d1_creatinine_min 2813
temp 2735
wbc 2709
creatinine 2700
calcium 2694
d1_creatinine_max 2692
d1_platelets_max 2635
d1_hematocrit_max 2631
heart_rate_apache 2603
h1_heartrate_max 2602
d1_wbc_min 2595
d1_heartrate_min 2586
d1_hematocrit_min 2585
bun 2577
sysbp 2571
d1_temp_max 2536
heartrate 2504
resprate 2502
hco3 2493
h1_sysbp_min 2467
potassium 2466
d1_sysbp_noninvasive_max 2463
bun_apache 2446
d1_hemaglobin_max 2442
hematocrit 2425
diasbp 2411
temp_apache 2404
d1_sysbp_noninvasive_min 2392
hemaglobin 2375
d1_calcium_min 2369
h1_heartrate_min 2365
h1_sysbp_noninvasive_min 2335
h1_sysbp_max 2325
sodium 2314
h1_diasbp_min 2276
d1_bun_min 2272
d1_calcium_max 2252
d1_hco3_min 2244
height 2214
h1_diasbp_noninvasive_min 2204
h1_diasbp_max 2194
d1_heartrate_max 2193
resprate_apache 2192
d1_mbp_noninvasive_max 2168
d1_hemaglobin_min 2159
h1_temp_max 2143
h1_temp_min 2141
h1_sysbp_noninvasive_max 2127
po2 2116
map_apache 2112
d1_temp_min 2109
d1_diasbp_max 2109
d1_diasbp_noninvasive_max 2107
h1_diasbp_noninvasive_max 2086
h1_mbp_noninvasive_max 2082
h1_resprate_min 2068
d1_potassium_max 2057
d1_diasbp_noninvasive_min 2055
d1_bun_max 2042
h1_mbp_noninvasive_min 2033
sodium_apache 2012
h1_resprate_max 2007
mbp 1997
d1_mbp_noninvasive_min 1966
d1_hco3_max 1963
d1_sysbp_max 1958
d1_resprate_max 1956
spo2 1954
d1_resprate_min 1942
d1_diasbp_min 1939
d1_potassium_min 1898
d1_sysbp_min 1884
h1_mbp_max 1825
albumin 1782
h1_mbp_min 1762
d1_mbp_min 1741
d1_sodium_min 1726
d1_mbp_max 1723
d1_arterial_pco2_min 1612
icu_type 1593
d1_sodium_max 1578
d1_arterial_po2_min 1575
lactate 1572
d1_inr_max 1554
bilirubin 1531
eatures


d1_spo2_min 1510
d1_arterial_pco2_max 1485
ph 1470
d1_arterial_po2_max 1464
d1_sysbp_invasive_max 1450
d1_lactate_max 1428
d1_inr_min 1422
h1_spo2_min 1400
d1_diasbp_invasive_min 1398
d1_arterial_ph_min 1394
d1_albumin_max 1388
pco2 1384
d1_arterial_ph_max 1381
d1_lactate_min 1357
inr 1340
albumin_apache 1285
h1_spo2_max 1231
d1_albumin_min 1226
hospital_admit_source 1209
d1_mbp_invasive_max 1198
d1_sysbp_invasive_min 1192
d1_bilirubin_max 1169
d1_pao2fio2ratio_min 1167
d1_diasbp_invasive_max 1155
d1_pao2fio2ratio_max 1144
bilirubin_apache 1138
h1_inr_max 1105
d1_bilirubin_min 1097
pao2fio2ratio 1070
h1_inr_min 1053
pao2_apache 1030
h1_sysbp_invasive_max 1008
d1_mbp_invasive_min 1002
ethnicity 995
paco2_apache 921
h1_creatinine_max 910
h1_hematocrit_max 909
ph_apache 896
h1_arterial_po2_min 861
h1_diasbp_invasive_min 850
h1_hematocrit_min 840
h1_potassium_max 834
h1_bun_max 827
h1_diasbp_invasive_max 796
h1_mbp_invasive_min 791
h1_hemaglobin_min 787
h1_hemaglobin_max 772
h1_wbc_max 763
h1_bun_min 762
h1_potassium_min 749
paco2_for_ph_apache 745
h1_creatinine_min 744
h1_calcium_max 735
h1_sysbp_invasive_min 719
h1_arterial_pco2_min 703
d1_spo2_max 702
h1_mbp_invasive_max 700
h1_arterial_po2_max 691
h1_calcium_min 688
h1_platelets_max 686
h1_hco3_max 684
h1_wbc_min 668
fio2_apache 651
h1_platelets_min 650
gcs_motor_apache 633
h1_arterial_pco2_max 632
gcs_eyes_apache 577
h1_pao2fio2ratio_max 575
h1_sodium_min 575
h1_arterial_ph_max 571
h1_arterial_ph_min 567
gcs_verbal_apache 546
arf_apache 545
h1_hco3_min 539
h1_sodium_max 523
h1_pao2fio2ratio_min 515
h1_lactate_max 442
h1_lactate_min 326
icu_admit_source 305
gender 259
h1_albumin_max 243
icu_stay_type 242
h1_albumin_min 236
cirrhosis 230
h1_bilirubin_max 195
h1_bilirubin_min 189
gcs_unable_apache 84
ventilated_apache 45
hepatic_failure 39
apache_post_operative 34
elective_surgery 28
intubated_apache 20
leukemia 9
solid_tumor_with_metastasis 6
immunosuppression 4
lymphoma 1
0
1000
2000
3000
4000
5000
Feature importance

```
# drop columns with more than 85% NA
columns_to_drop=X_train.isna().sum()[X_train.isna().sum() > len(X_train)*0.85] #columns with more than 85% NA
```

```
columns_to_drop.index.to_list()
```

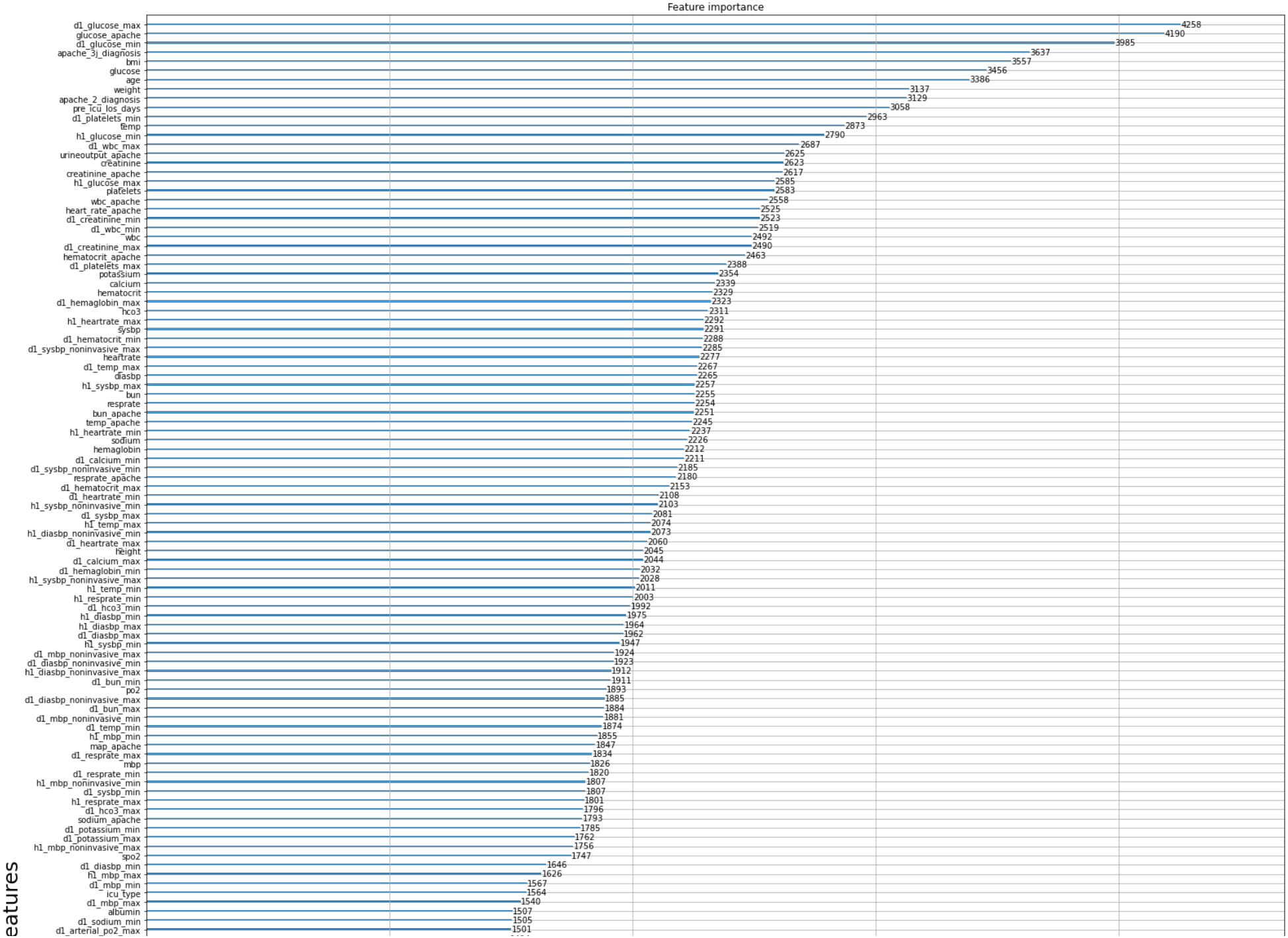
```
['h1_albumin_max',
 'h1_albumin_min',
 'h1_bilirubin_max',
 'h1_bilirubin_min',
 'h1_lactate_max',
 'h1_lactate_min',
```

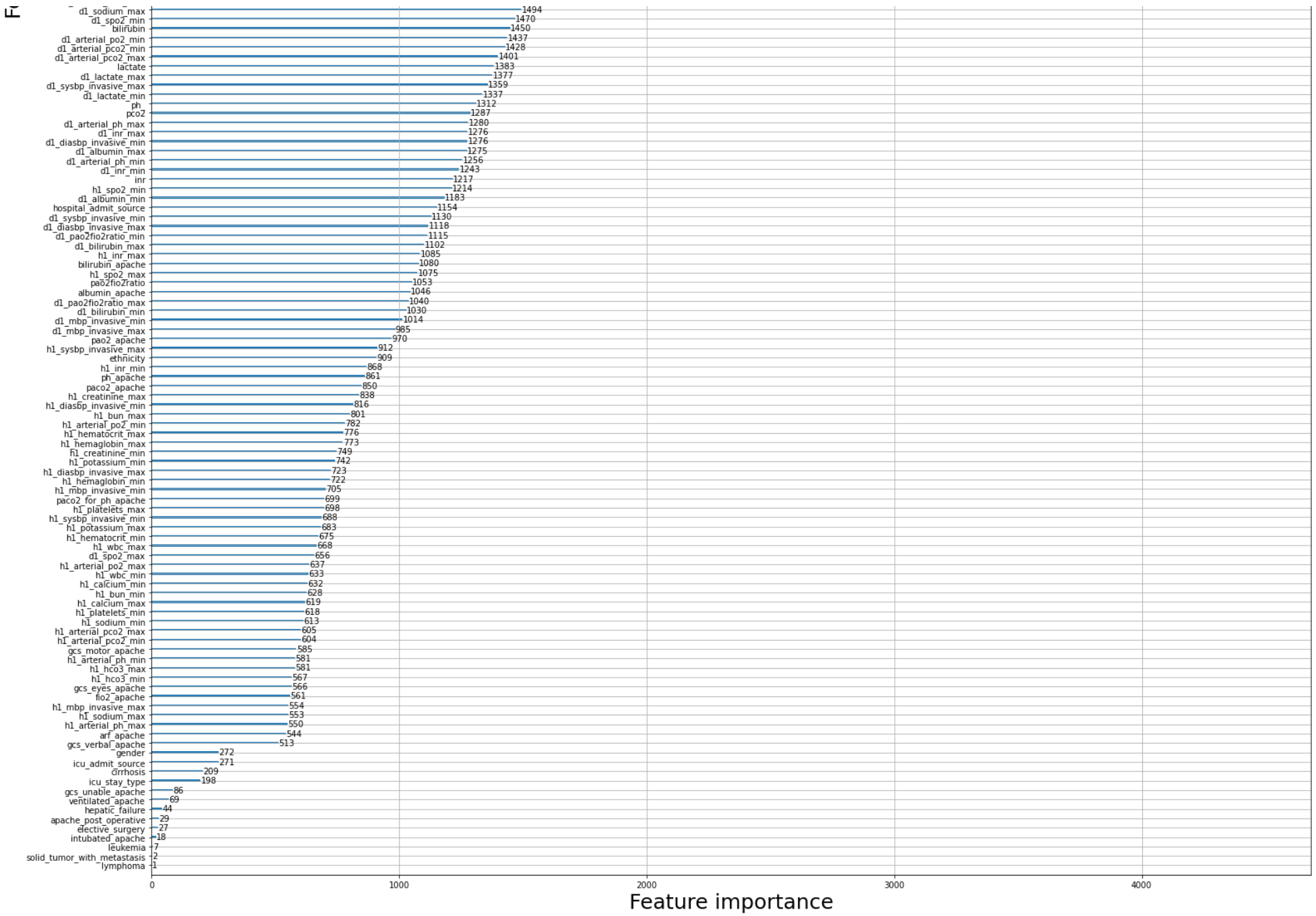
```
      'h1_pao2fio2ratio_max',
      'h1_pao2fio2ratio_min']

X_train=X_train.drop(columns_to_drop.index.to_list(), axis=1)
X_test=X_test.drop(columns_to_drop.index.to_list(), axis=1)


#let´s do it again
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.824411
[200]   test's auc: 0.831034
[300]   test's auc: 0.837748
[400]   test's auc: 0.842611
[500]   test's auc: 0.846707
[600]   test's auc: 0.849376
[700]   test's auc: 0.851246
[800]   test's auc: 0.852889
[900]   test's auc: 0.854257
[1000]  test's auc: 0.855396
[1100]  test's auc: 0.85623
[1200]  test's auc: 0.85693
[1300]  test's auc: 0.857491
[1400]  test's auc: 0.857973
[1500]  test's auc: 0.858306
[1600]  test's auc: 0.858616
[1700]  test's auc: 0.858914
[1800]  test's auc: 0.859092
[1900]  test's auc: 0.85933
[2000]  test's auc: 0.859511
[2100]  test's auc: 0.859636
[2200]  test's auc: 0.859823
[2300]  test's auc: 0.859904
[2400]  test's auc: 0.859976
[2500]  test's auc: 0.860059
[2600]  test's auc: 0.860067
[2700]  test's auc: 0.86012
[2800]  test's auc: 0.860119
[2900]  test's auc: 0.860134
```

```
[3000]  test's auc: 0.860187
[3100]  test's auc: 0.860183
[3200]  test's auc: 0.860179
Early stopping, best iteration is:
[3012]  test's auc: 0.860202
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=100,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
lgb.plot_importance(clf_6, figsize = (25,40))
plt.show()
```


Feature importance
d1_glucose_max 4258
glucose_apache 4190
d1_glucose_min 3985
apache_3j_diagnosis 3637
bmi 3557
glucose 3456
age 3386
weight 3137
apache_2_diagnosis 3129
pre_icu_los_days 3058
d1_platelets_min 2963
temp 2873
h1_glucose_min 2790
d1_wbc_max 2687
urineoutput_apache 2625
creatinine 2623
creatinine_apache 2617
h1_glucose_max 2585
platelets 2583
wbc_apache 2558
heart_rate_apache 2525
d1_creatinine_min 2523
d1_wbc_min 2519
wbc 2492
d1_creatinine_max 2490
hematocrit_apache 2463
d1_platelets_max 2388
potassium 2354
calcium 2339
hematocrit 2329
d1_hemaglobin_max 2323
hco3 2311
h1_heartrate_max 2292
sysbp 2291
d1_hematocrit_min 2288
d1_sysbp_noninvasive_max 2285
heartrate 2277
d1_temp_max 2267
diasbp 2265
h1_sysbp_max 2257
bun 2255
resprate 2254
bun_apache 2251
temp_apache 2245
h1_heartrate_min 2237
sodium 2226
hemaglobin 2212
d1_calcium_min 2211
d1_sysbp_noninvasive_min 2185
resprate_apache 2180
d1_hematocrit_max 2153
d1_heartrate_min 2108
h1_sysbp_noninvasive_min 2103
d1_sysbp_max 2081
h1_temp_max 2074
h1_diasbp_noninvasive_min 2073
d1_heartrate_max 2060
height 2045
d1_calcium_max 2044
d1_hemaglobin_min 2032
h1_sysbp_noninvasive_max 2028
h1_temp_min 2011
h1_resprate_min 2003
d1_hco3_min 1992
h1_diasbp_min 1975
h1_diasbp_max 1964
d1_diasbp_max 1962
h1_sysbp_min 1947
d1_mbp_noninvasive_max 1924
d1_diasbp_noninvasive_min 1923
h1_diasbp_noninvasive_max 1912
d1_bun_min 1911
po2 1893
d1_diasbp_noninvasive_max 1885
d1_bun_max 1884
d1_mbp_noninvasive_min 1881
d1_temp_min 1874
h1_mbp_min 1855
map_apache 1847
d1_resprate_max 1834
mbp 1826
d1_resprate_min 1820
h1_mbp_noninvasive_min 1807
d1_sysbp_min 1807
h1_resprate_max 1801
d1_hco3_max 1796
sodium_apache 1793
d1_potassium_min 1785
d1_potassium_max 1762
h1_mbp_noninvasive_max 1756
spo2 1747
d1_diasbp_min 1646
h1_mbp_max 1626
d1_mbp_min 1567
icu_type 1564
d1_mbp_max 1540
albumin 1507
d1_sodium_min 1505
d1_arterial_po2_max 1501
eatures


d1_sodium_max 1494
d1_spo2_min 1470
bilirubin 1450
d1_arterial_po2_min 1437
d1_arterial_pco2_min 1428
d1_arterial_pco2_max 1401
lactate 1383
d1_lactate_max 1377
d1_sysbp_invasive_max 1359
d1_lactate_min 1337
ph 1312
pco2 1287
d1_arterial_ph_max 1280
d1_inr_max 1276
d1_diasbp_invasive_min 1276
d1_albumin_max 1275
d1_arterial_ph_min 1256
d1_inr_min 1243
inr 1217
h1_spo2_min 1214
d1_albumin_min 1183
hospital_admit_source 1154
d1_sysbp_invasive_min 1130
d1_diasbp_invasive_max 1118
d1_pao2fio2ratio_min 1115
d1_bilirubin_max 1102
h1_inr_max 1085
bilirubin_apache 1080
h1_spo2_max 1075
pao2fio2ratio 1053
albumin_apache 1046
d1_pao2fio2ratio_max 1040
d1_bilirubin_min 1030
d1_mbp_invasive_min 1014
d1_mbp_invasive_max 985
pao2_apache 970
h1_sysbp_invasive_max 912
ethnicity 909
h1_inr_min 868
ph_apache 861
paco2_apache 850
h1_creatinine_max 838
h1_diasbp_invasive_min 816
h1_bun_max 801
h1_arterial_po2_min 782
h1_hematocrit_max 776
h1_hemaglobin_max 773
h1_creatinine_min 749
h1_potassium_min 742
h1_diasbp_invasive_max 723
h1_hemaglobin_min 722
h1_mbp_invasive_min 705
paco2_for_ph_apache 699
h1_platelets_max 698
h1_sysbp_invasive_min 688
h1_potassium_max 683
h1_hematocrit_min 675
h1_wbc_max 668
d1_spo2_max 656
h1_arterial_po2_max 637
h1_wbc_min 633
h1_calcium_min 632
h1_bun_min 628
h1_calcium_max 619
h1_platelets_min 618
h1_sodium_min 613
h1_arterial_pco2_max 605
h1_arterial_pco2_min 604
gcs_motor_apache 585
h1_arterial_ph_min 581
h1_hco3_max 581
h1_hco3_min 567
gcs_eyes_apache 566
fio2_apache 561
h1_mbp_invasive_max 554
h1_sodium_max 553
h1_arterial_ph_max 550
arf_apache 544
gcs_verbal_apache 513
gender 272
icu_admit_source 271
cirrhosis 209
icu_stay_type 198
gcs_unable_apache 86
ventilated_apache 69
hepatic_failure 44
apache_post_operative 29
elective_surgery 27
intubated_apache 18
leukemia 7
solid_tumor_with_metastasis 2
lymphoma 1
0
1000
2000
3000
4000
Feature importance

```
#pre_icu_los_days seems to be important...
X_train['pre_icu_los_days'].describe()
```

```
count    91109.000000
mean         0.843242
std          2.506742
min         -0.244444
25%          0.045833
50%          0.155556
75%          0.420833
max        175.627778
Name: pre_icu_los_days, dtype: float64
```

```
#drop the least relevant ones, importance below 500 in the plot above
not_relevant=['gender', 'icu_admit_source', 'cirrhosis', 'icu_stay_type', 'gcs_unable_apache',
              'ventilated_apache', 'hepatic_failure', 'apache_post_operative','elective_surgery',
              'intubated_apache', 'leukemia', 'solid_tumor_with_metastasis', 'lymphoma', 'immunosuppression' ]
```

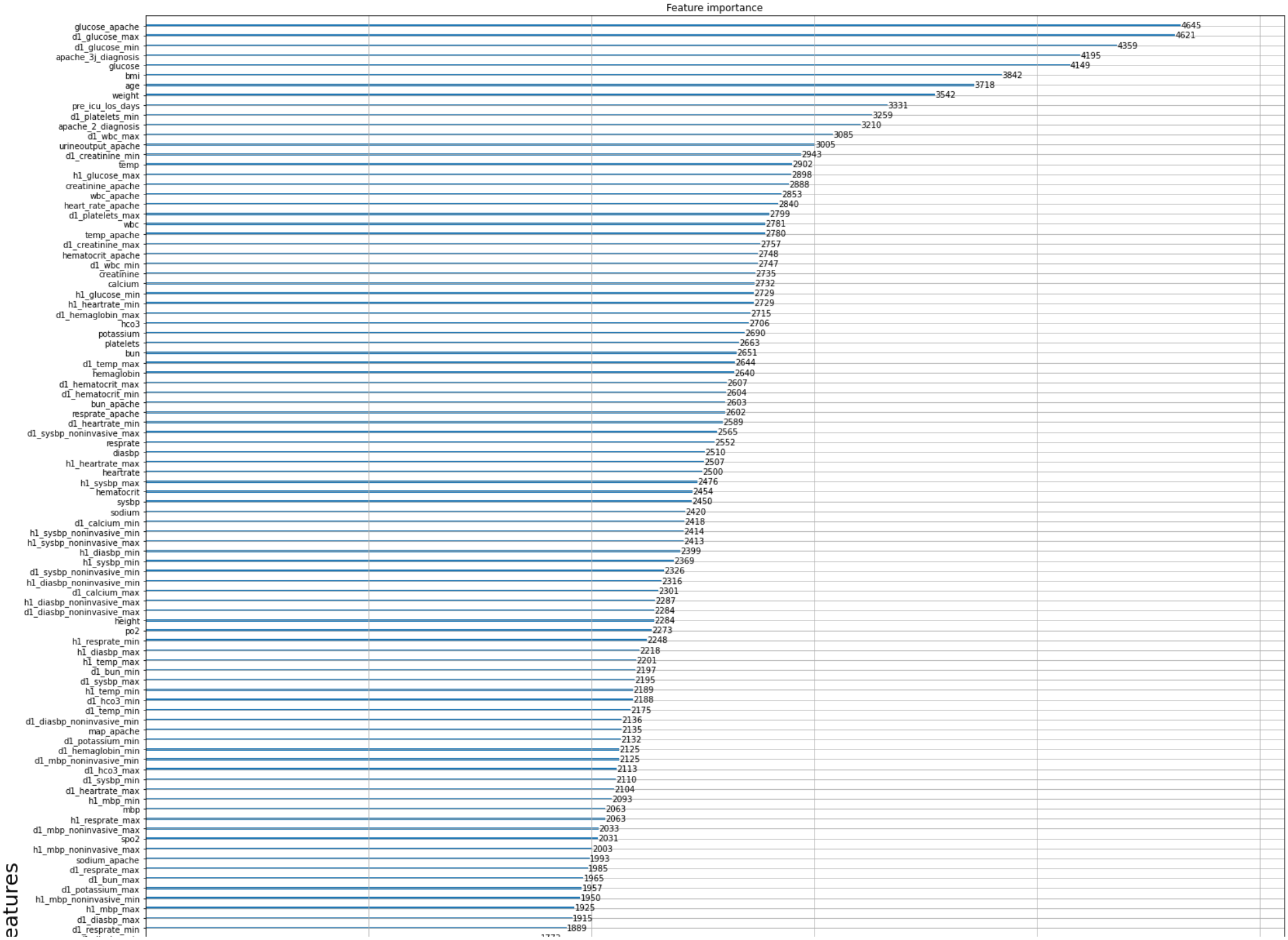
```
X_train=X_train.drop(not_relevant, axis=1)
X_test=X_test.drop(not_relevant, axis=1)
```

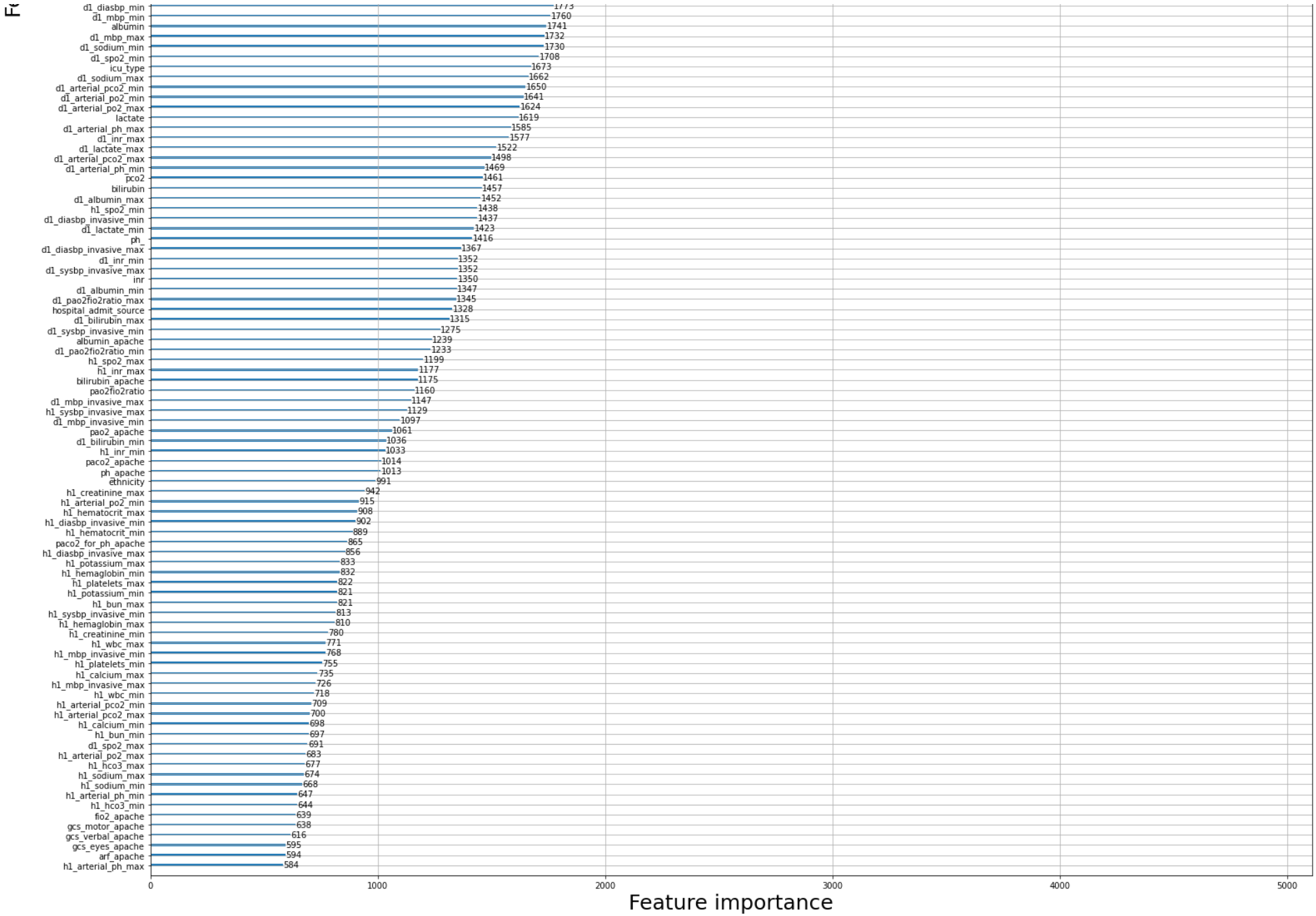
```
#let´s do it once again
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.824196
[200]   test's auc: 0.832443
[300]   test's auc: 0.838266
[400]   test's auc: 0.84261
[500]   test's auc: 0.845914
[600]   test's auc: 0.848925
```

```
[700]	test's auc: 0.850962
[800]	test's auc: 0.852632
[900]	test's auc: 0.854027
[1000]	test's auc: 0.855093
[1100]	test's auc: 0.856008
[1200]	test's auc: 0.856807
[1300]	test's auc: 0.857363
[1400]	test's auc: 0.85775
[1500]	test's auc: 0.858069
[1600]	test's auc: 0.8584
[1700]	test's auc: 0.858677
[1800]	test's auc: 0.858865
[1900]	test's auc: 0.859017
[2000]	test's auc: 0.859144
[2100]	test's auc: 0.859359
[2200]	test's auc: 0.859467
[2300]	test's auc: 0.859547
[2400]	test's auc: 0.859648
[2500]	test's auc: 0.859733
[2600]	test's auc: 0.859759
[2700]	test's auc: 0.859822
[2800]	test's auc: 0.859851
[2900]	test's auc: 0.859902
[3000]	test's auc: 0.859908
[3100]	test's auc: 0.859923
[3200]	test's auc: 0.85992
[3300]	test's auc: 0.859969
[3400]	test's auc: 0.859972
[3500]	test's auc: 0.859959
[3600]	test's auc: 0.85994
Early stopping, best iteration is:
[3371]	test's auc: 0.85999
LGBMClassifier(colsample_bytree=0.2, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=100,
               objective='binary', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=0.5, subsample=1)
```

```
lgb.plot_importance(clf_6, figsize = (25,40))
plt.show()
```


Feature importance
glucose_apache 4645
d1_glucose_max 4621
d1_glucose_min 4359
apache_3j_diagnosis 4195
glucose 4149
bmi 3842
age 3718
weight 3542
pre_icu_los_days 3331
d1_platelets_min 3259
apache_2_diagnosis 3210
d1_wbc_max 3085
urineoutput_apache 3005
d1_creatinine_min 2943
temp 2902
h1_glucose_max 2898
creatinine_apache 2888
wbc_apache 2853
heart_rate_apache 2840
d1_platelets_max 2799
wbc 2781
temp_apache 2780
d1_creatinine_max 2757
hematocrit_apache 2748
d1_wbc_min 2747
creatinine 2735
calcium 2732
h1_glucose_min 2729
h1_heartrate_min 2729
d1_hemaglobin_max 2715
hco3 2706
potassium 2690
platelets 2663
bun 2651
d1_temp_max 2644
hemaglobin 2640
d1_hematocrit_max 2607
d1_hematocrit_min 2604
bun_apache 2603
resprate_apache 2602
d1_heartrate_min 2589
d1_sysbp_noninvasive_max 2565
resprate 2552
diasbp 2510
h1_heartrate_max 2507
heartrate 2500
h1_sysbp_max 2476
hematocrit 2454
sysbp 2450
sodium 2420
d1_calcium_min 2418
h1_sysbp_noninvasive_min 2414
h1_sysbp_noninvasive_max 2413
h1_diasbp_min 2399
h1_sysbp_min 2369
d1_sysbp_noninvasive_min 2326
h1_diasbp_noninvasive_min 2316
d1_calcium_max 2301
h1_diasbp_noninvasive_max 2287
d1_diasbp_noninvasive_max 2284
height 2284
po2 2273
h1_resprate_min 2248
h1_diasbp_max 2218
h1_temp_max 2201
d1_bun_min 2197
d1_sysbp_max 2195
h1_temp_min 2189
d1_hco3_min 2188
d1_temp_min 2175
d1_diasbp_noninvasive_min 2136
map_apache 2135
d1_potassium_min 2132
d1_hemaglobin_min 2125
d1_mbp_noninvasive_min 2125
d1_hco3_max 2113
d1_sysbp_min 2110
d1_heartrate_max 2104
h1_mbp_min 2093
mbp 2063
h1_resprate_max 2063
d1_mbp_noninvasive_max 2033
spo2 2031
h1_mbp_noninvasive_max 2003
sodium_apache 1993
d1_resprate_max 1985
d1_bun_max 1965
d1_potassium_max 1957
h1_mbp_noninvasive_min 1950
h1_mbp_max 1925
d1_diasbp_max 1915
d1_resprate_min 1889
eatures

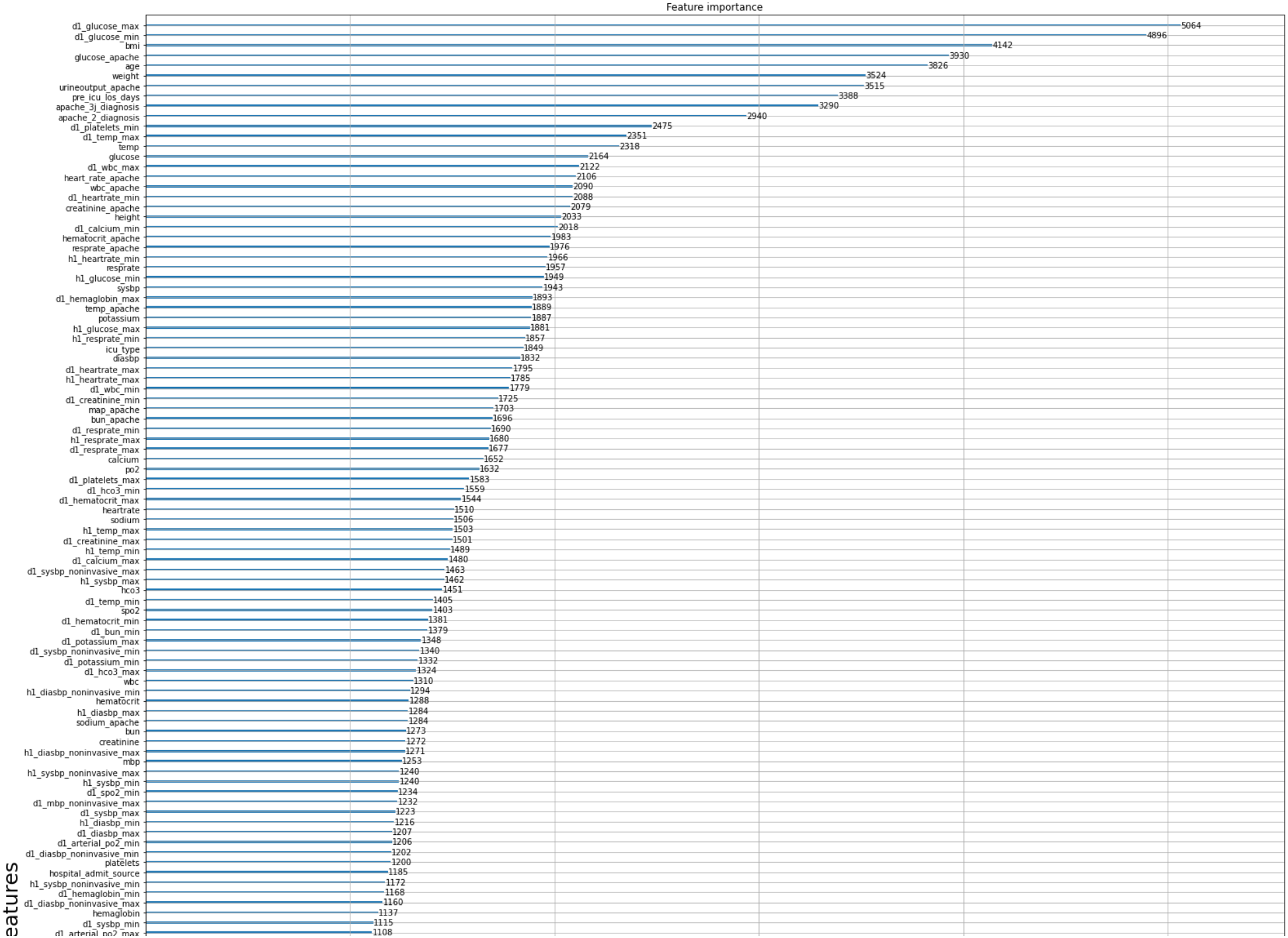

d1_diasbp_min 1773
d1_mbp_min 1760
albumin 1741
d1_mbp_max 1732
d1_sodium_min 1730
d1_spo2_min 1708
icu_type 1673
d1_sodium_max 1662
d1_arterial_pco2_min 1650
d1_arterial_po2_min 1641
d1_arterial_po2_max 1624
lactate 1619
d1_arterial_ph_max 1585
d1_inr_max 1577
d1_lactate_max 1522
d1_arterial_pco2_max 1498
d1_arterial_ph_min 1469
pco2 1461
bilirubin 1457
d1_albumin_max 1452
h1_spo2_min 1438
d1_diasbp_invasive_min 1437
d1_lactate_min 1423
ph_ 1416
d1_diasbp_invasive_max 1367
d1_inr_min 1352
d1_sysbp_invasive_max 1352
inr 1350
d1_albumin_min 1347
d1_pao2fio2ratio_max 1345
hospital_admit_source 1328
d1_bilirubin_max 1315
d1_sysbp_invasive_min 1275
albumin_apache 1239
d1_pao2fio2ratio_min 1233
h1_spo2_max 1199
h1_inr_max 1177
bilirubin_apache 1175
pao2fio2ratio 1160
d1_mbp_invasive_max 1147
h1_sysbp_invasive_max 1129
d1_mbp_invasive_min 1097
pao2_apache 1061
d1_bilirubin_min 1036
h1_inr_min 1033
paco2_apache 1014
ph_apache 1013
ethnicity 991
h1_creatinine_max 942
h1_arterial_po2_min 915
h1_hematocrit_max 908
h1_diasbp_invasive_min 902
h1_hematocrit_min 889
paco2_for_ph_apache 865
h1_diasbp_invasive_max 856
h1_potassium_max 833
h1_hemaglobin_min 832
h1_platelets_max 822
h1_potassium_min 821
h1_bun_max 821
h1_sysbp_invasive_min 813
h1_hemaglobin_max 810
h1_creatinine_min 780
h1_wbc_max 771
h1_mbp_invasive_min 768
h1_platelets_min 755
h1_calcium_max 735
h1_mbp_invasive_max 726
h1_wbc_min 718
h1_arterial_pco2_min 709
h1_arterial_pco2_max 700
h1_calcium_min 698
h1_bun_min 697
d1_spo2_max 691
h1_arterial_po2_max 683
h1_hco3_max 677
h1_sodium_max 674
h1_sodium_min 668
h1_arterial_ph_min 647
h1_hco3_min 644
fio2_apache 639
gcs_motor_apache 638
gcs_verbal_apache 616
gcs_eyes_apache 595
arf_apache 594
h1_arterial_ph_max 584
0
1000
2000
3000
4000
5000
Feature importance

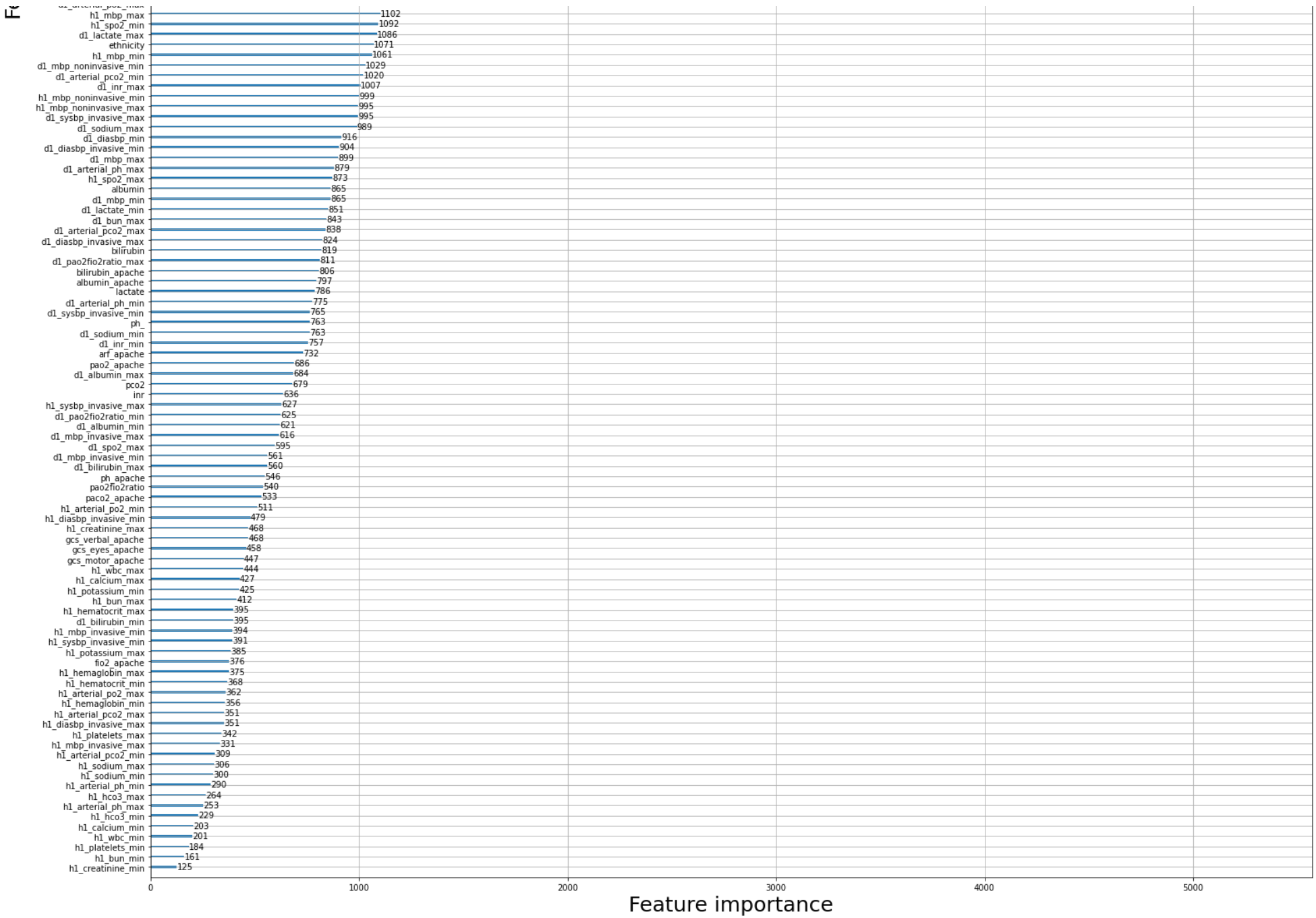
```
#import lightgbm as lgb

clf_6 = lgb.LGBMClassifier(boosting_type='gbdt', objective='cross_entropy',n_jobs=-1,
                                 metric= 'auc', scale_pos_weight= 2, subsample = 1, force_col_wise = True,
                                learning_rate= 0.01, colsample_bytree= 1, reg_alpha= 3, reg_lambda= 1,
                                n_estimators=5000, max_depth=-1, num_leaves=100) #changed colsample_bytree to 1 and scale_pos_weight to
```

```
#let´s do it once again
clf_6.fit(X_train, y_train, early_stopping_rounds= 240, eval_metric= 'auc',
                            eval_set= [(X_test,y_test)], eval_names= ['test'], verbose= 100, feature_name= 'auto', categorical_featur
```

```
Training until validation scores don't improve for 240 rounds
[100]   test's auc: 0.834413
[200]   test's auc: 0.841154
[300]   test's auc: 0.846027
[400]   test's auc: 0.84944
[500]   test's auc: 0.852109
[600]   test's auc: 0.853768
[700]   test's auc: 0.855003
[800]   test's auc: 0.855807
[900]   test's auc: 0.856274
[1000]  test's auc: 0.856571
[1100]  test's auc: 0.8568
[1200]  test's auc: 0.856894
[1300]  test's auc: 0.857002
[1400]  test's auc: 0.857072
[1500]  test's auc: 0.85712
[1600]  test's auc: 0.857169
[1700]  test's auc: 0.857191
[1800]  test's auc: 0.857208
[1900]  test's auc: 0.857234
[2000]  test's auc: 0.857257
[2100]  test's auc: 0.857259
[2200]  test's auc: 0.857305
[2300]  test's auc: 0.857285
[2400]  test's auc: 0.857296
Early stopping, best iteration is:
[2206]  test's auc: 0.857319
LGBMClassifier(colsample_bytree=1, force_col_wise=True, learning_rate=0.01,
               metric='auc', n_estimators=5000, num_leaves=100,
               objective='cross_entropy', reg_alpha=3, reg_lambda=1,
               scale_pos_weight=2, subsample=1)
```

```
lgb.plot_importance(clf_6, figsize = (25,40))
plt.show()
```


Feature importance
d1_glucose_max 5064
d1_glucose_min 4896
bmi 4142
glucose_apache 3930
age 3826
weight 3524
urineoutput_apache 3515
pre_icu_los_days 3388
apache_3j_diagnosis 3290
apache_2_diagnosis 2940
d1_platelets_min 2475
d1_temp_max 2351
temp 2318
glucose 2164
d1_wbc_max 2122
heart_rate_apache 2106
wbc_apache 2090
d1_heartrate_min 2088
creatinine_apache 2079
height 2033
d1_calcium_min 2018
hematocrit_apache 1983
resprate_apache 1976
h1_heartrate_min 1966
resprate 1957
h1_glucose_min 1949
sysbp 1943
d1_hemaglobin_max 1893
temp_apache 1889
potassium 1887
h1_glucose_max 1881
h1_resprate_min 1857
icu_type 1849
diasbp 1832
d1_heartrate_max 1795
h1_heartrate_max 1785
d1_wbc_min 1779
d1_creatinine_min 1725
map_apache 1703
bun_apache 1696
d1_resprate_min 1690
h1_resprate_max 1680
d1_resprate_max 1677
calcium 1652
po2 1632
d1_platelets_max 1583
d1_hco3_min 1559
d1_hematocrit_max 1544
heartrate 1510
sodium 1506
h1_temp_max 1503
d1_creatinine_max 1501
h1_temp_min 1489
d1_calcium_max 1480
d1_sysbp_noninvasive_max 1463
h1_sysbp_max 1462
hco3 1451
d1_temp_min 1405
spo2 1403
d1_hematocrit_min 1381
d1_bun_min 1379
d1_potassium_max 1348
d1_sysbp_noninvasive_min 1340
d1_potassium_min 1332
d1_hco3_max 1324
wbc 1310
h1_diasbp_noninvasive_min 1294
hematocrit 1288
h1_diasbp_max 1284
sodium_apache 1284
bun 1273
creatinine 1272
h1_diasbp_noninvasive_max 1271
mbp 1253
h1_sysbp_noninvasive_max 1240
h1_sysbp_min 1240
d1_spo2_min 1234
d1_mbp_noninvasive_max 1232
d1_sysbp_max 1223
h1_diasbp_min 1216
d1_diasbp_max 1207
d1_arterial_po2_min 1206
d1_diasbp_noninvasive_min 1202
platelets 1200
hospital_admit_source 1185
h1_sysbp_noninvasive_min 1172
d1_hemaglobin_min 1168
d1_diasbp_noninvasive_max 1160
hemaglobin 1137
d1_sysbp_min 1115
d1_arterial_po2_max 1108
eatures


h1_mbp_max 1102
h1_spo2_min 1092
d1_lactate_max 1086
ethnicity 1071
h1_mbp_min 1061
d1_mbp_noninvasive_min 1029
d1_arterial_pco2_min 1020
d1_inr_max 1007
h1_mbp_noninvasive_min 999
h1_mbp_noninvasive_max 995
d1_sysbp_invasive_max 995
d1_sodium_max 989
d1_diasbp_min 916
d1_diasbp_invasive_min 904
d1_mbp_max 899
d1_arterial_ph_max 879
h1_spo2_max 873
albumin 865
d1_mbp_min 865
d1_lactate_min 851
d1_bun_max 843
d1_arterial_pco2_max 838
d1_diasbp_invasive_max 824
bilirubin 819
d1_pao2fio2ratio_max 811
bilirubin_apache 806
albumin_apache 797
lactate 786
d1_arterial_ph_min 775
d1_sysbp_invasive_min 765
ph_ 763
d1_sodium_min 763
d1_inr_min 757
arf_apache 732
pao2_apache 686
d1_albumin_max 684
pco2 679
inr 636
h1_sysbp_invasive_max 627
d1_pao2fio2ratio_min 625
d1_albumin_min 621
d1_mbp_invasive_max 616
d1_spo2_max 595
d1_mbp_invasive_min 561
d1_bilirubin_max 560
ph_apache 546
pao2fio2ratio 540
paco2_apache 533
h1_arterial_po2_min 511
h1_diasbp_invasive_min 479
h1_creatinine_max 468
gcs_verbal_apache 468
gcs_eyes_apache 458
gcs_motor_apache 447
h1_wbc_max 444
h1_calcium_max 427
h1_potassium_min 425
h1_bun_max 412
h1_hematocrit_max 395
d1_bilirubin_min 395
h1_mbp_invasive_min 394
h1_sysbp_invasive_min 391
h1_potassium_max 385
fio2_apache 376
h1_hemaglobin_max 375
h1_hematocrit_min 368
h1_arterial_po2_max 362
h1_hemaglobin_min 356
h1_arterial_pco2_max 351
h1_diasbp_invasive_max 351
h1_platelets_max 342
h1_mbp_invasive_max 331
h1_arterial_pco2_min 309
h1_sodium_max 306
h1_sodium_min 300
h1_arterial_ph_min 290
h1_hco3_max 264
h1_arterial_ph_max 253
h1_hco3_min 229
h1_calcium_min 203
h1_wbc_min 201
h1_platelets_min 184
h1_bun_min 161
h1_creatinine_min 125
0
1000
2000
3000
4000
5000
Feature importance

imput to use different models

```
from missMDA.svdtriplet import svdtriplet
from missMDA.imputePCA import imputePCA
from missMDA.estim_ncpPCA import estim_ncpPCA
```

```
selection_main2=['bun','albumin', 'creatinine',  'lactate', 'bilirubin','hematocrit', 'hemaglobin', 'sysbp',
                    'mbp', 'diasbp','sodium', 'potassium', 'hco3', 'glucose', 'pco2', 'po2', 'calcium',
                    'heartrate', 'inr', 'pao2fio2ratio', 'platelets', 'resprate', 'spo2', 'temp', 'wbc', 'age', 'weight', 'height', '
```

```
X_train2=X_train[selection_main2]
X_test2=X_test[selection_main2]
```

```
estim_ncpPCA(X_train2, ncpmax=10)
```

```
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
Stopped after criterion < threshold
[0,
 [762.578082405118,
  797.6708176307113,
  767.5651704945195,
  795.2054815575683,
  835.8466369270287,
  838.4871244584682,
  905.108203047842,
  790.4797588607687,
  846.4006003048037,
  914.3651778653731,
  972.2686540823322]]
```

```
PCAinputed_averaged = imputePCA(X_train2, ncp=2) #Im gonna use 2
```

```
Stopped after criterion < threshold
```

```
X_train2_imputedDF=pd.DataFrame(PCAinputed_averaged[0])
```