**TRAIN DATA INFO**

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

RangeIndex: 7464 entries, 0 to 7463

Data columns (total 12 columns):

index 7464 non-null int64

Maximum Degree 7464 non-null float64

Minimum Degree 7464 non-null float64

Molecular Weight 7464 non-null float64

Number of H-Bond Donors 7464 non-null float64

Number of Rings 7464 non-null float64

Number of Rotatable Bonds 7464 non-null float64

Polar Surface Area 7464 non-null float64

inchi\_key 7464 non-null object

Graph 7464 non-null object

smiles 7464 non-null object

target 7464 non-null int64

dtypes: float64(7), int64(2), object(3)

memory usage: 699.8+ KB

**TEST DATA INFO**

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

RangeIndex: 1867 entries, 0 to 1866

Data columns (total 11 columns):

index 1867 non-null int64

Maximum Degree 1867 non-null float64

Minimum Degree 1867 non-null float64

Molecular Weight 1867 non-null float64

Number of H-Bond Donors 1867 non-null float64

Number of Rings 1867 non-null float64

Number of Rotatable Bonds 1867 non-null float64

Polar Surface Area 1867 non-null float64

inchi\_key 1867 non-null object

Graph 1867 non-null object

smiles 1867 non-null object

dtypes: float64(7), int64(1), object(3)

memory usage: 160.5+ KB

**TRAIN DATA describe**

index Maximum Degree Minimum Degree Molecular Weight \

count 7464.000000 7464.000000 7464.000000 7464.000000

mean 3731.500000 3.288183 0.875804 278.941072

std 2154.815537 0.556245 0.390472 169.849479

min 0.000000 0.000000 0.000000 41.053000

25% 1865.750000 3.000000 1.000000 164.248000

50% 3731.500000 3.000000 1.000000 240.444000

75% 5597.250000 4.000000 1.000000 346.467000

max 7463.000000 4.000000 2.000000 1950.681000

Number of H-Bond Donors Number of Rings Number of Rotatable Bonds \

count 7464.000000 7464.000000 7464.000000

mean 1.252412 1.728162 4.208199

std 2.012638 1.679037 4.525664

min 0.000000 0.000000 0.000000

25% 0.000000 1.000000 1.000000

50% 1.000000 1.000000 3.000000

75% 2.000000 3.000000 6.000000

max 36.000000 30.000000 47.000000

Polar Surface Area target

count 7464.000000 7464.000000

mean 60.600695 0.040729

std 62.155964 0.197675

min 0.000000 0.000000

25% 26.300000 0.000000

50% 46.530000 0.000000

75% 77.430000 0.000000

max 1095.850000 1.000000

inchi\_key Graph \

count 7464 7464

unique 7464 5086

top NCGC00259117-01 [((0; 1); 1.0); ((1; 2); 1.5); ((2; 3); 1.5); ...

freq 1 28

smiles

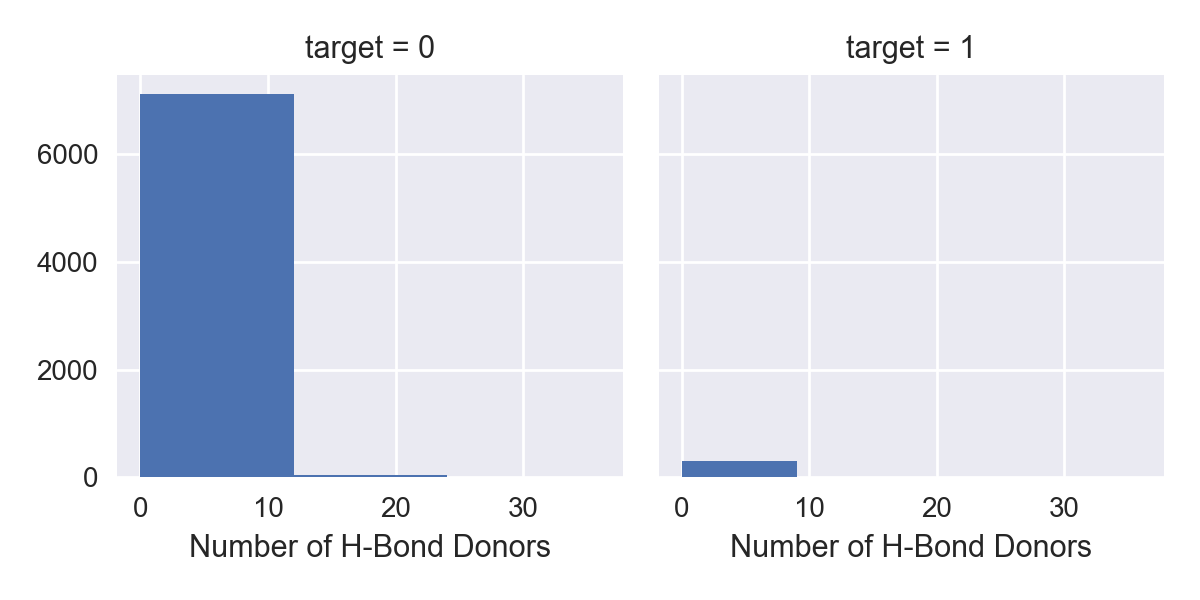
count 7464

unique 6215

top CC1=CC=C(O)C=C1

freq 5

https://www.kaggle.com/saxinou/imbalanced-data-xgboost-tunning



HBond\_Band target

0 (-0.036, 6.0] 0.040120

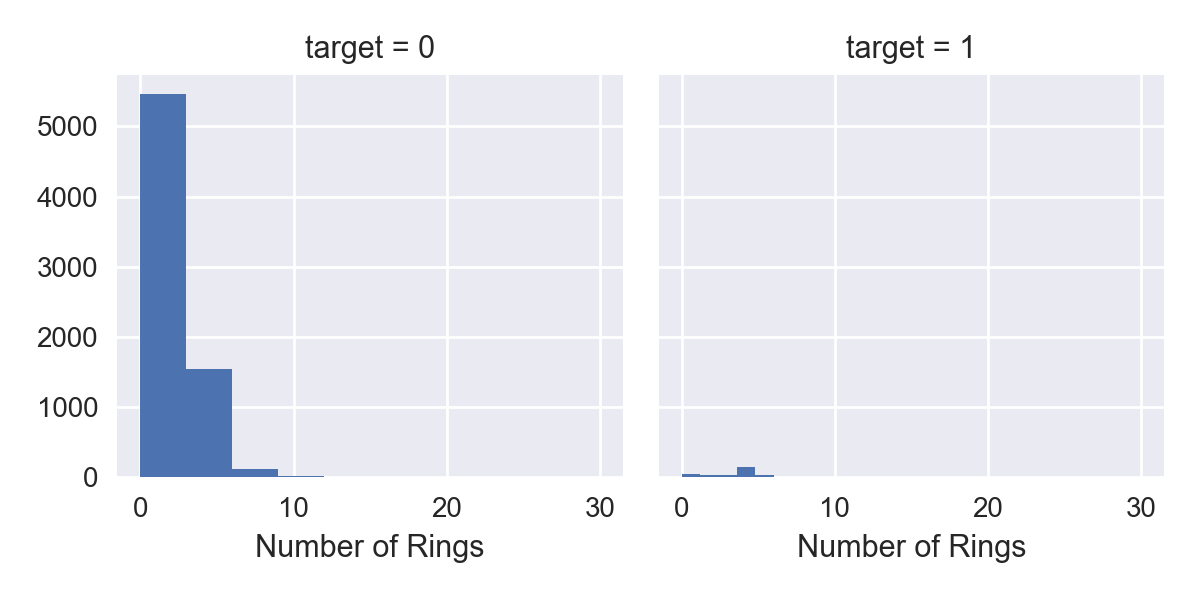
1 (6.0, 12.0] 0.073684

2 (12.0, 18.0] 0.034483

3 (18.0, 24.0] 0.142857

4 (24.0, 30.0) 0.500000

5 (30.0, 36.0) 0.000000



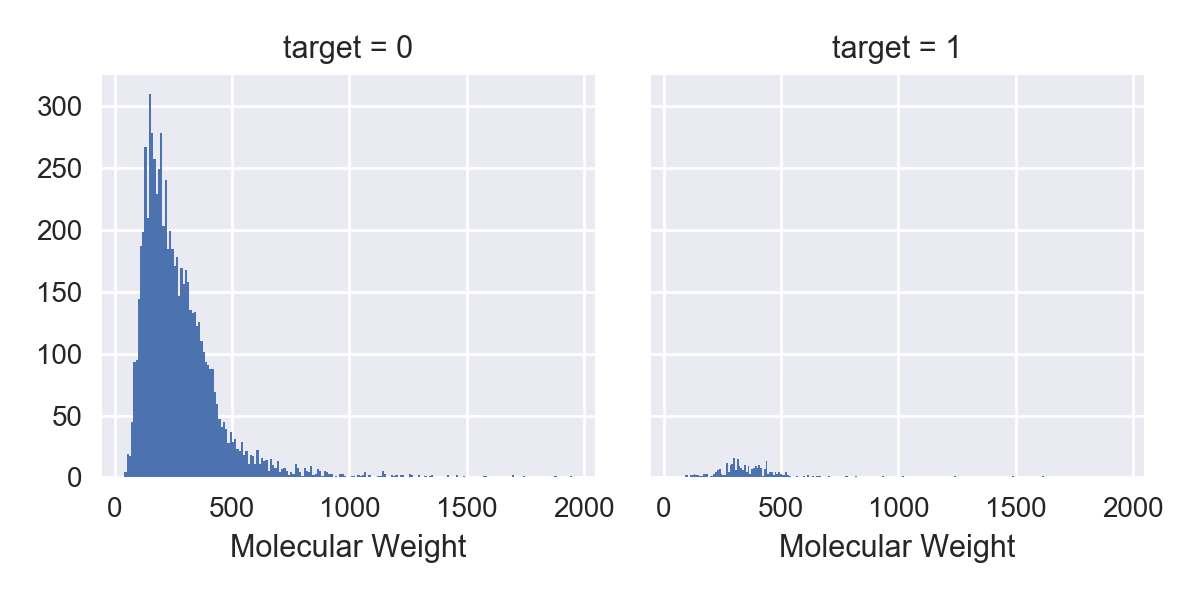
Ring\_Band target

0 (-0.03, 7.5] 0.040621

1 (7.5, 15.0] 0.062500

2 (15.0, 22.5) 0.000000

3 (22.5, 30.0) 0.000000



WeightBand target

0 (39.143, 359.324] 0.029086

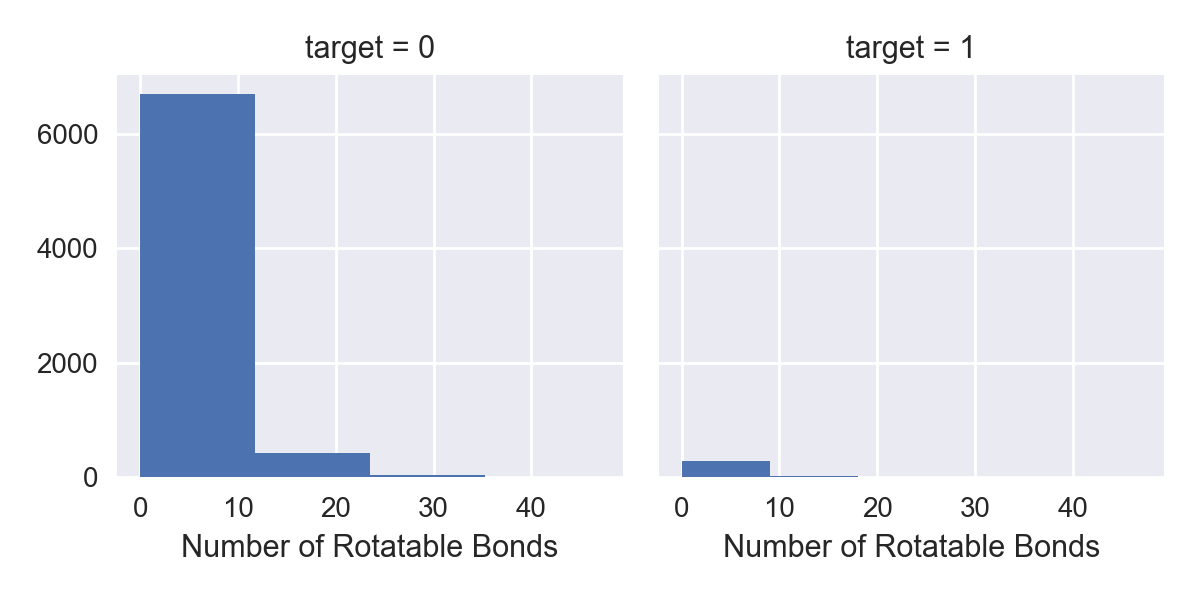
1 (359.324, 677.596] 0.086277

2 (677.596, 995.867] 0.031847

3 (995.867, 1314.138] 0.048780

4 (1314.138, 1632.41) 0.153846

5 (1632.41, 1950.681) 0.000000



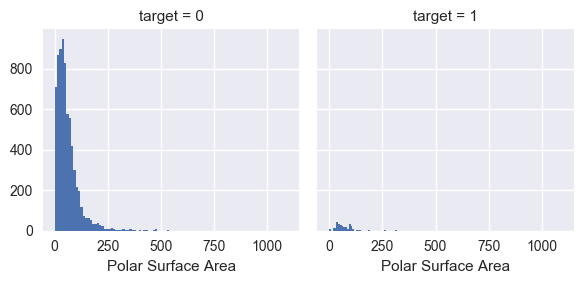
Rotate\_Band target

0 (-0.047, 11.75] 0.041523

1 (11.75, 23.5] 0.027714

2 (23.5, 35.25) 0.027778

3 (35.25, 47.0) 0.090909



Area\_Band target

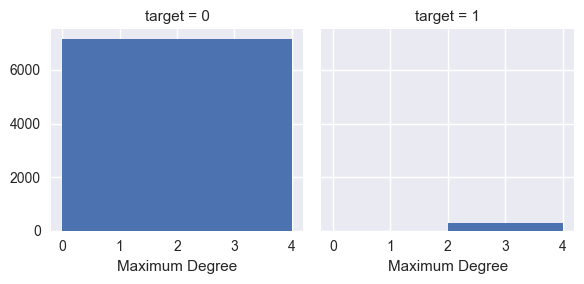
0 (-1.096, 219.17] 0.039951

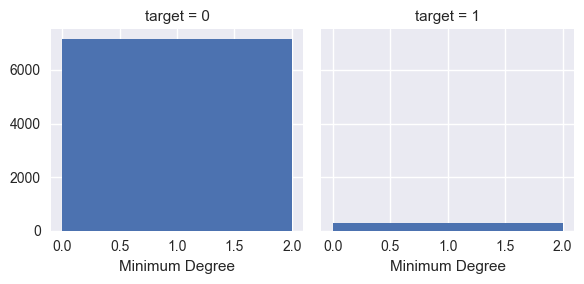
1 (219.17, 438.34] 0.080000

2 (438.34, 657.51] 0.045455

3 (657.51, 876.68) 0.200000

4 (876.68, 1095.85) 0.000000





[((0; 1); 1.0); ((1; 2); 1.5); ((2; 3); 1.5); ((3; 4); 1.0); ((3; 5); 1.5); ((5; 6); 1.5); ((6; 1); 1.5)]

<class 'str'>

CC1=CC(N)=NO1

<class 'str'>

10-fold:

**2-max 2 min(best)**

Mean and std AUC score for train set: 0.999715844595 3.23006881932e-05

Mean and std AUC score for test set: 0.842769793356 0.0480596904455

tunning

Mean and std AUC score for train set: 0.903485800382 0.00343935676942

Mean and std AUC score for test set: 0.855101294528 0.0495548343651

**0.01 5000**

Mean and std AUC score for train set: 0.932376811945 0.00217777309598

Mean and std AUC score for test set: 0.856186024509 0.0464228986061

{'learning\_rate': 0.005} 0.854001970223

+ density

Mean and std AUC score for train set: 0.939691796793 0.00227589039292

Mean and std AUC score for test set: 0.856354673515 0.0510060011438

+mean degree

Mean and std AUC score for train set: 0.936135020393 0.00227617995778

Mean and std AUC score for test set: 0.852471616507 0.0428728580612

0.023 5000

Mean and std AUC score for train set: 0.947854181096 0.00161150140768

Mean and std AUC score for test set: 0.85455930498 0.0455790230333

0.001 1000

Mean and std AUC score for train set: 0.869332990575 0.00540107231028

Mean and std AUC score for test set: 0.842427764762 0.0522848106973

5000 0.001

Mean and std AUC score for train set: 0.892534360752 0.00390321096777

Mean and std AUC score for test set: 0.85268456779 0.0517915509267

2 max 1 min no 'Min\_Vertex\_betweenness'

Mean and std AUC score for train set: 0.999716412201 3.20560774351e-05

Mean and std AUC score for test set: 0.840483645702 0.0470392516037

Tunning

Mean and std AUC score for train set: 0.905149169329 0.0030315663681

Mean and std AUC score for test set: 0.855990343605 0.0500192972152

2 max no 2 min

Mean and std AUC score for train set: 0.999716865706 3.21634976788e-05

Mean and std AUC score for test set: 0.834717291404 0.0524400364071

**After adding 2 min and average, LB score goes down from 0.85925 to 0.84052. CV results seem better but LB went down.**

**After adding 1 min, and tunning, still 0.84052. Weird.**

0.85925: No tunning, just 2 max.

**Test 4.13 night**

**# xgb = XGBRegressor(learning\_rate=0.01, n\_estimators=1000, max\_depth=6,**

**# min\_child\_weight=6, gamma=0, subsample=0.6, colsample\_bytree=0.9,**

**# objective='binary:logistic', nthread=4, scale\_pos\_weight=weight\_balance,**

**# seed=27, reg\_alpha=118)**

+ density, 2 max\_betweenness

Mean and std AUC score for train set: 0.900433263099 0.00402588697663

Mean and std AUC score for test set: 0.849387952784 0.0593472702037

+ 2 max\_betweenness

Mean and std AUC score for train set: 0.894068851908 0.00412248354674

Mean and std AUC score for test set: 0.849030005406 0.0577073183888

+2 max + path avg

Mean and std AUC score for train set: 0.899356379468 0.00389328429829

Mean and std AUC score for test set: 0.85154434733 0.0569255982025

+ 2 max 2 min

Mean and std AUC score for train set: 0.903485800382 0.00343935676942

Mean and std AUC score for test set: 0.855101294528 0.0495548343651

**+2 max 1 min**

Mean and std AUC score for train set: 0.905149169329 0.0030315663681

Mean and std AUC score for test set: 0.855990343605 0.0500192972152

+2 max 1 min + density

Mean and std AUC score for train set: 0.904140619732 0.00348449708037

Mean and std AUC score for test set: 0.855102495945 0.0500291413568

**2max 1 min avg-path**

Mean and std AUC score for train set: 0.907853583346 0.00347223139932

Mean and std AUC score for test set: 0.856660133958 0.0499357732801

2max 1 min avg-path density

Mean and std AUC score for train set: 0.908859200933 0.00322599845894

Mean and std AUC score for test set: 0.855946566949 0.0511016576032

+2 max 1 min + diameter

Mean and std AUC score for train set: 0.905881270813 0.003640993595

Mean and std AUC score for test set: 0.855422974109 0.050520127341

+2 max 1 min + is\_simple

Mean and std AUC score for train set: 0.903498970056 0.00344443116337

Mean and std AUC score for test set: 0.855204165916 0.0492739172055

+2 max 1 min + #edges

Mean and std AUC score for train set: 0.904215677392 0.00363273758052

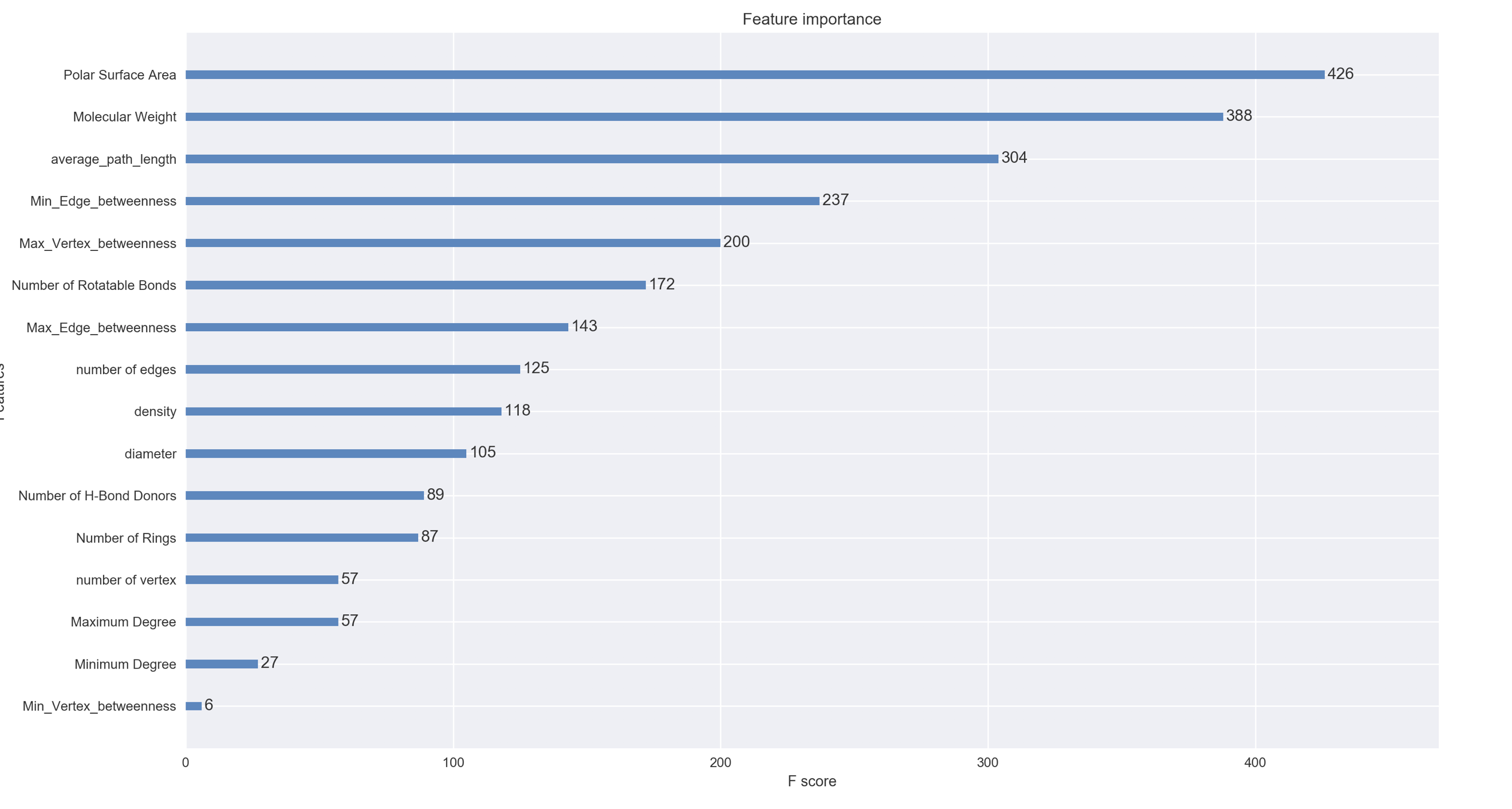
Mean and std AUC score for test set: 0.855931023608 0.0489369238508

+2 max 1 min + #vertex

Mean and std AUC score for train set: 0.904470463621 0.0036280188282

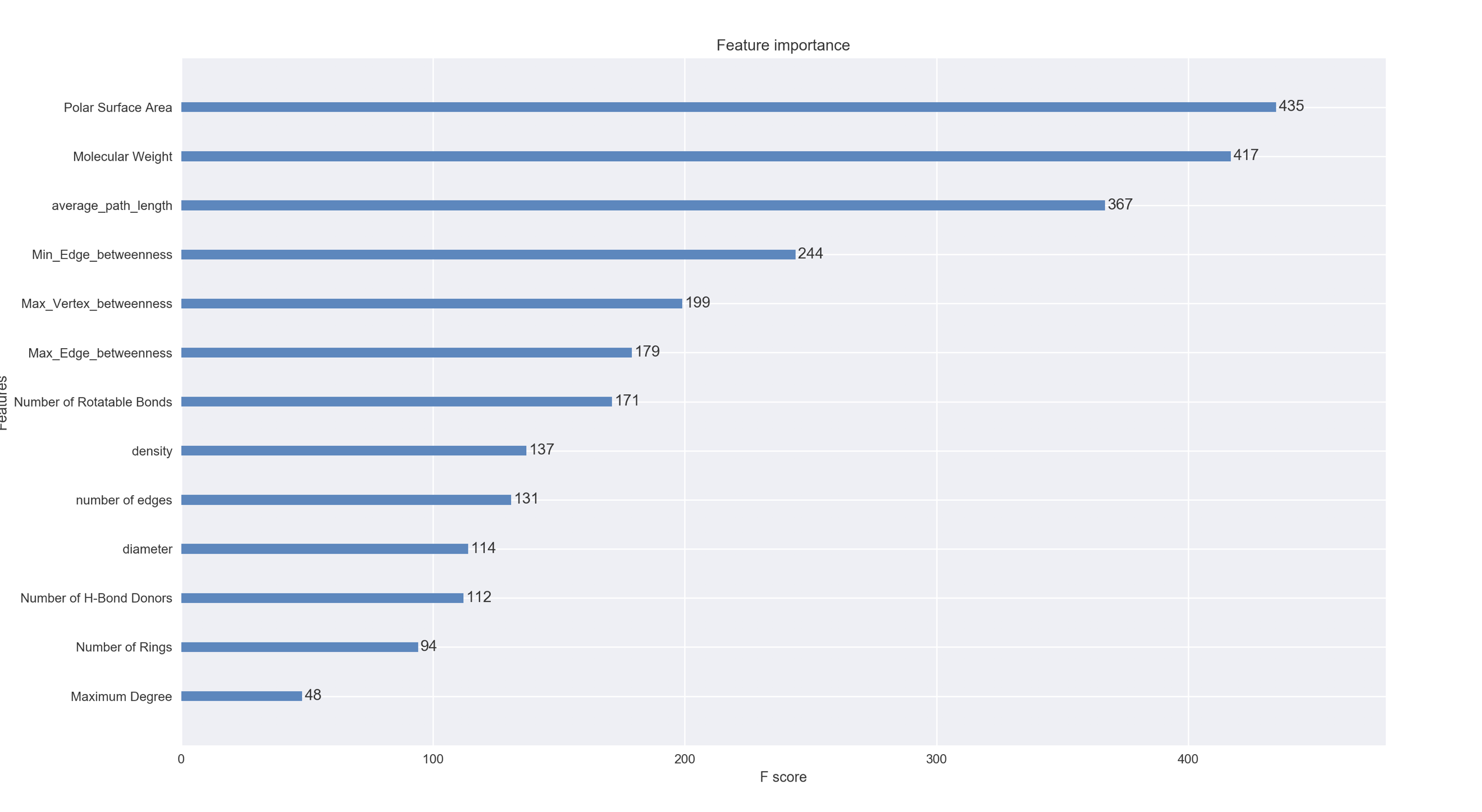
Mean and std AUC score for test set: 0.85591044933 0.0497228591016

Feature selection:



Mean and std AUC score for train set: 0.928175066948 0.00445230583996

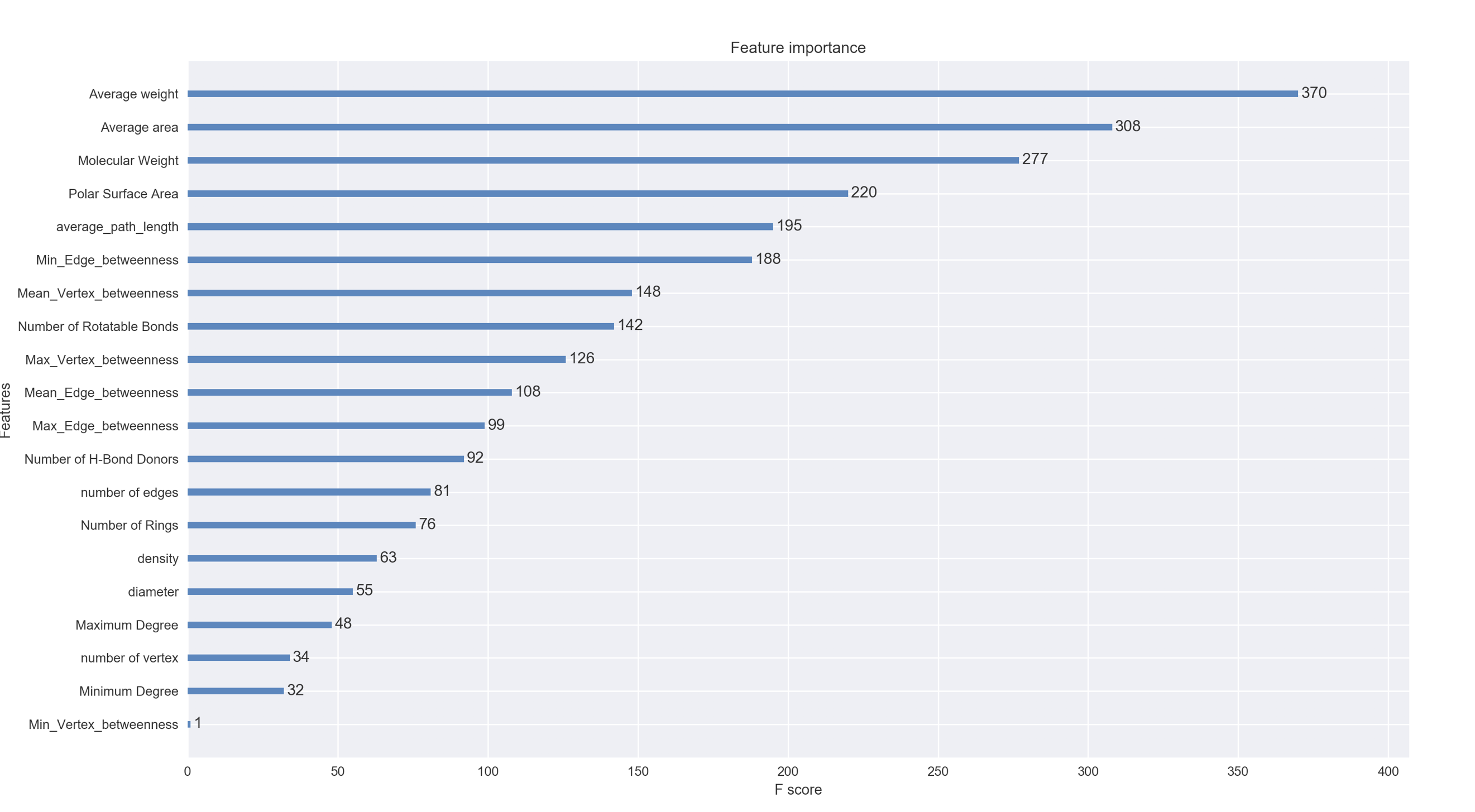
Mean and std AUC score for test set: 0.835930573076 0.0446284826436



Mean and std AUC score for train set: 0.931070570067 0.0036704179038

Mean and std AUC score for test set: 0.833414053583 0.0394600870001

查看feature importance的反馈是非常重要的，比如本场比赛制胜的关键是运用manager\_id这个feature，而它的feature importance反馈结果也是非常高。通过对重要特征的重新再提取特征，能够发现很多有意思的新特征，这才是用FE打好一场比赛的关键所在。



Mean and std AUC score for train set: 0.935148758894 0.00410882306397

Mean and std AUC score for test set: 0.844306857091 0.0541140909614

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness',

'Minimum Degree', 'number of vertex',

'number of edges',

'diameter', # after drop performance increase

# 'density',

# 'average\_path\_length',

# 'Average area', 'Average weight',

# 'Mean\_Vertex\_betweenness','Mean\_Edge\_betweenness'

]

Mean and std AUC score for train set: 0.933806275777 0.00518856381033

Mean and std AUC score for test set: 0.850562188382 0.0466737075043

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness']

Mean and std AUC score for train set: 0.935901393998 0.00418956304846

Mean and std AUC score for test set: 0.844827746741 0.0559542449841

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness',

'Minimum Degree',

Mean and std AUC score for train set: 0.936027023716 0.00395007613762

Mean and std AUC score for test set: 0.845407956389 0.0550979304134

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness',

'Minimum Degree',

# 'number of vertex',

**'number of edges'**,]

Mean and std AUC score for train set: 0.936008107856 0.0036743741684

Mean and std AUC score for test set: 0.84887051721 0.0535042291672

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness',

'Minimum Degree',

# 'number of vertex',

# 'number of edges',

'diameter', # after drop perofrmance increase

Mean and std AUC score for train set: 0.933300747306 0.00338306155095

Mean and std AUC score for test set: 0.854209392083 0.0509409055315

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness',

'Minimum Degree',

# 'number of vertex',

# 'number of edges',

'diameter', # after drop perofrmance increase

'density',

Mean and std AUC score for train set: 0.933376914869 0.0031501935796

Mean and std AUC score for test set: **0.855992370998** 0.0415633729718

**4.15 I got 0.856 on CV but 0.81 on LB**

drop\_list = ['inchi\_key', 'smiles', 'Graph', 'mol', 'igraph', 'Vertex\_betweenness',

'Edge\_betweenness','Min\_Vertex\_betweenness',

'Minimum Degree',

# 'number of vertex',

# 'number of edges',

'diameter', # after drop perofrmance increase

'density',

# 'Number of Rings',

'average\_path\_length',

Mean and std AUC score for train set: 0.934909264742 0.00417736121992

Mean and std AUC score for test set: **0.856128581726** 0.0450132623617

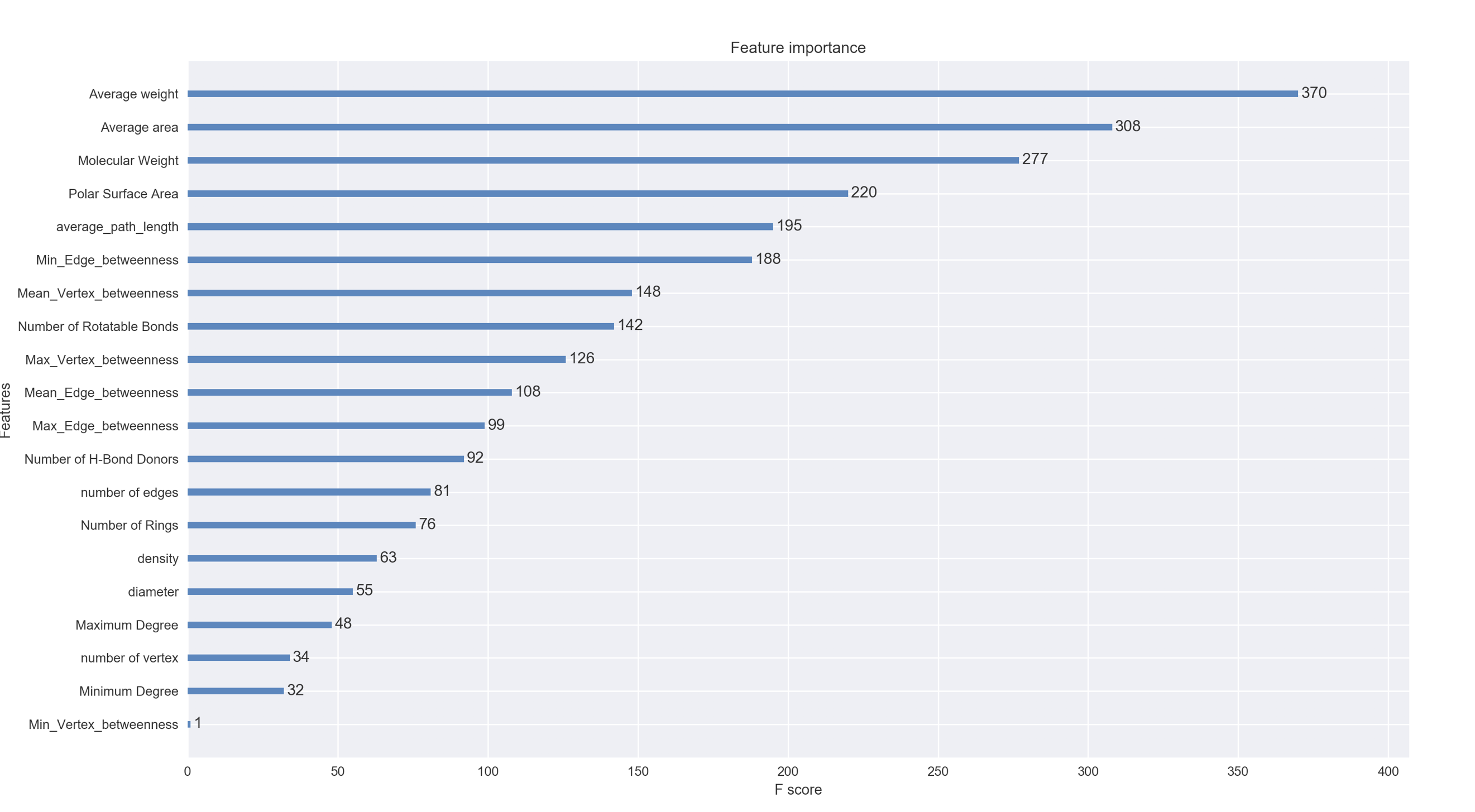
+inbalance

Mean and std AUC score for train set: 0.973796775069 0.00264635250851

Mean and std AUC score for test set: **0.84578722893** 0.0536685038943

++++++++++++Median\_Vertex\_betweenness Nothing. Std nothing

++++++++++++Median\_Edge\_betweenness Nothing.



+ 3 percentage

Mean and std AUC score for train set: 0.942911181547 0.00398709581828

Mean and std AUC score for test set: 0.837905328287 0.0453943073526

+Bond 3 percentage

Mean and std AUC score for train set: 0.941638810694 0.00352265635016

Mean and std AUC score for test set: 0.832368594942 0.0450946069588

+ Bond 2 percentage

Mean and std AUC score for train set: 0.940766344233 0.00529047636429

Mean and std AUC score for test set: 0.833471421277 0.0545011272851

+ Bond 1 percentage

Mean and std AUC score for train set: 0.943823012223 0.0054660933198

Mean and std AUC score for test set: 0.83711689794 0.0531225951409

+ Bond 1 num

Mean and std AUC score for train set: 0.934871330953 0.00492504303739

Mean and std AUC score for test set: 0.837738856851 0.0519371928829

+ Bond 2 num

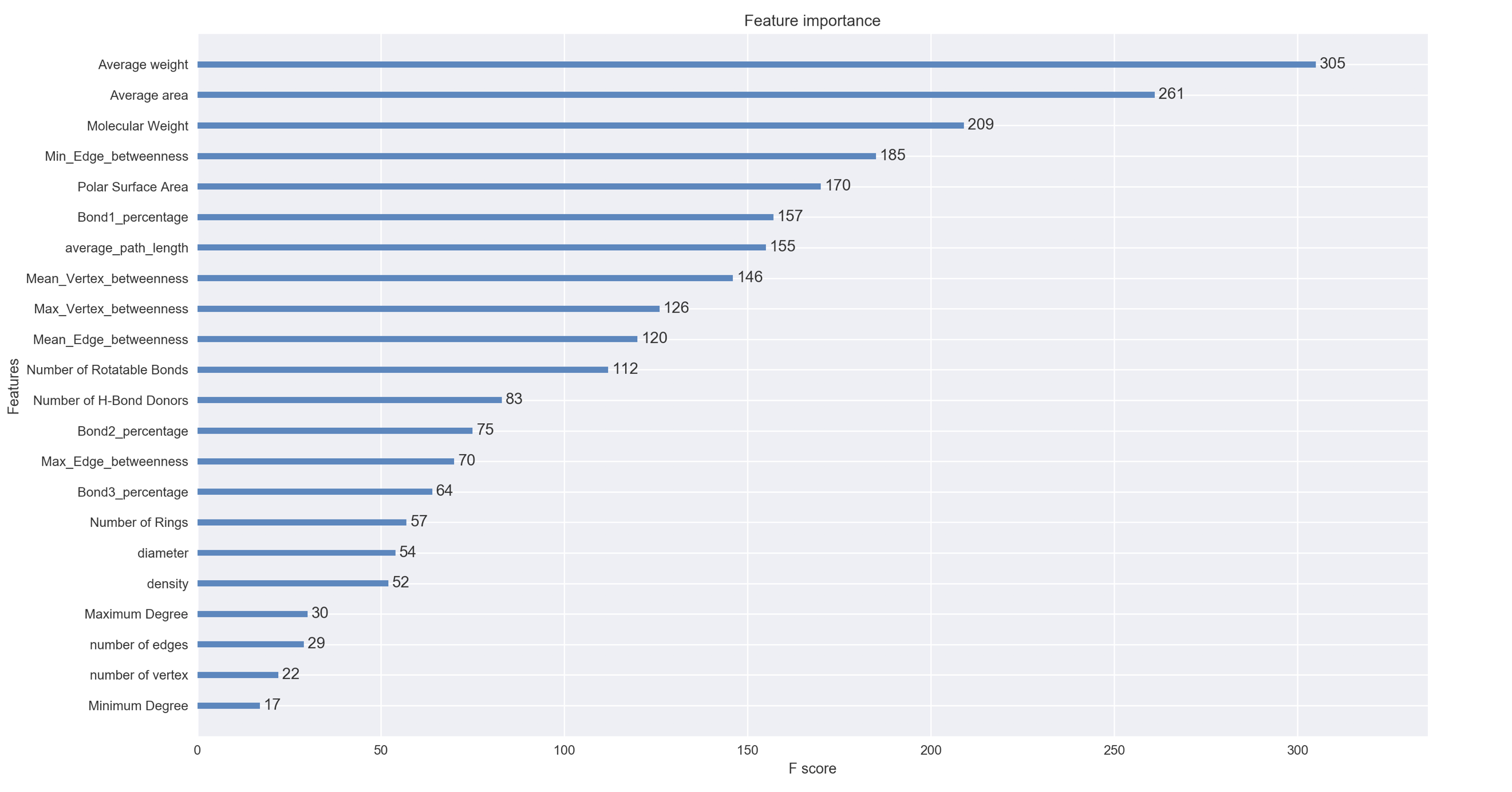
Mean and std AUC score for train set: 0.942031459951 0.00363936800304

Mean and std AUC score for test set: 0.84646362708 0.044049960115

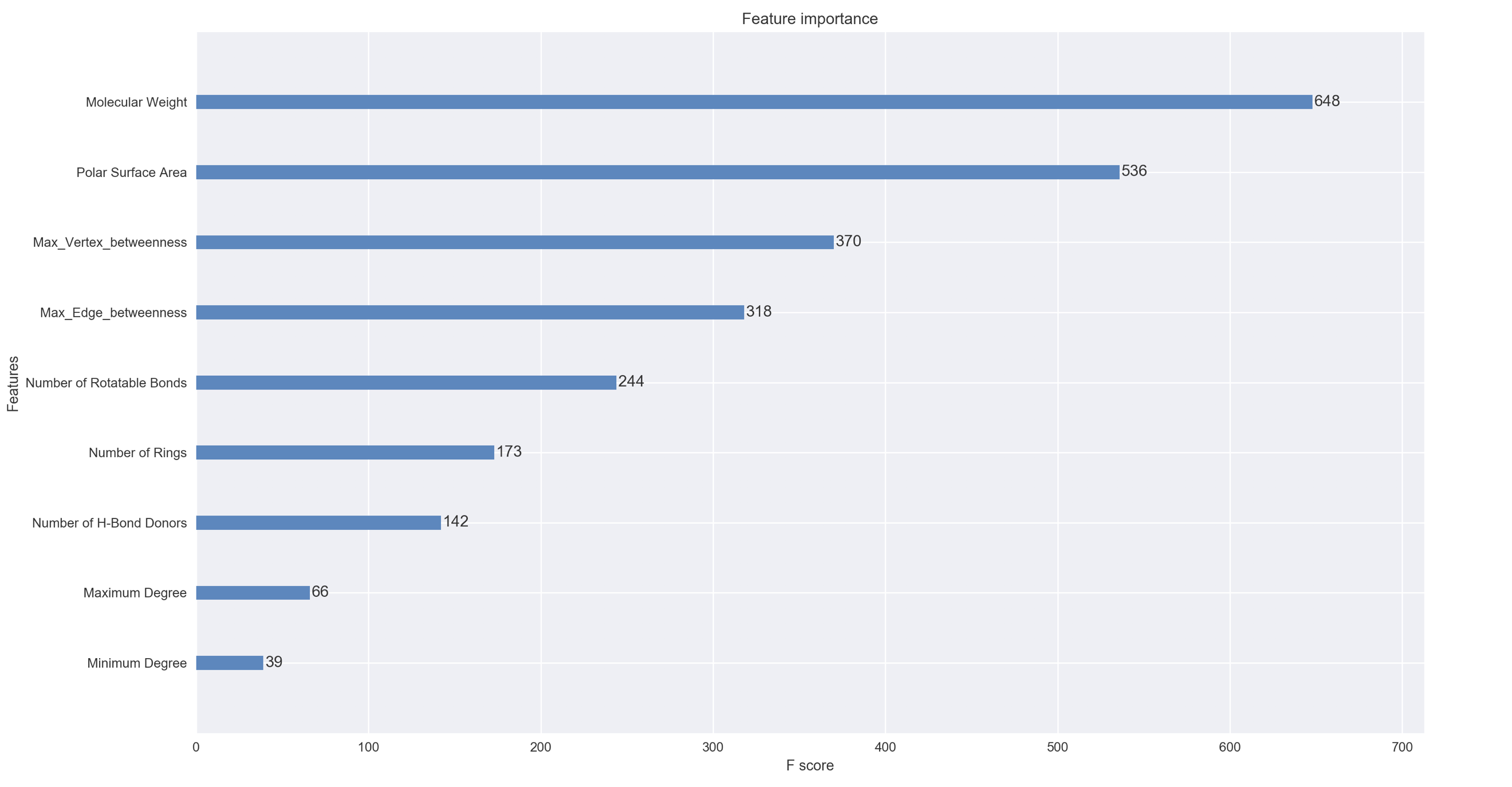
+ Bond 3 num

Mean and std AUC score for train set: 0.938497991723 0.00288042199681

Mean and std AUC score for test set: 0.837458476002 0.0444148144904



4.17----Make the model simple:(It should be the features with 0.86 score on LB)



CV score: Mean and std AUC score for train set: 0.997058790685 0.00041833828242

Mean and std AUC score for test set: 0.817763304398 0.0468500741004

LB score: 0.84137

Use average weight instead:

Mean and std AUC score for train set: 0.996958670844 0.000247707452918

Mean and std AUC score for test set: 0.818087948802 0.0438236407087

Use average area instead:

Mean and std AUC score for train set: 0.996541445513 0.000410508355961

Mean and std AUC score for test set: 0.823936263209 0.0483825788938

Use average area and weight instead:

Mean and std AUC score for train set: 0.996824888661 0.000395374384753

Mean and std AUC score for test set: 0.824602066679 0.0355620782738

Use average area instead +avg path

Mean and std AUC score for train set: 0.997627631011 0.000324319332182

Mean and std AUC score for test set: **0.826514697932** 0.0513894622933

Use average area and weight instead: +avg path

Mean and std AUC score for train set: 0.997541520409 0.000332714519574

Mean and std AUC score for test set: 0.82456314343 0.0434419053886

B + 3Bond

Mean and std AUC score for train set: 0.998393341236 0.000228863043338

Mean and std AUC score for test set: 0.827289474827 0.0325681729572

B+ Bond1

Mean and std AUC score for train set: 0.998322979661 0.000174617743757

Mean and std AUC score for test set: 0.834751277523 0.0434085527734

Test\_df:

Before dropping dup: (1867, 11)

After dropping dup: (1789, 11)

Train\_df:

Before dropping dup: (7464, 12)

After dropping dup: (**6215**, 12)

**+target**

Before dropping dup: (7464, 12)

After dropping dup: (**6248,** 12)

Mean similarity

Mean and std AUC score for train set: 0.99734714969 0.000254437273751

Mean and std AUC score for test set: 0.821996019006 0.0357336378101

421

[mean: 0.80774, std: 0.04045, params: {'max\_depth': 2, 'min\_child\_weight': 1}, mean: 0.80678, std: 0.04315, params: {'max\_depth': 2, 'min\_child\_weight': 3}, mean: 0.80921, std: 0.04222, params: {'max\_depth': 2, 'min\_child\_weight': 5}, mean: 0.80921, std: 0.03899, params: {'max\_depth': 2, 'min\_child\_weight': 7}, mean: 0.80167, std: 0.03788, params: {'max\_depth': 4, 'min\_child\_weight': 1}, mean: 0.80329, std: 0.04890, params: {'max\_depth': 4, 'min\_child\_weight': 3}, mean: 0.80688, std: 0.04014, params: {'max\_depth': 4, 'min\_child\_weight': 5}, mean: 0.80802, std: 0.04178, params: {'max\_depth': 4, 'min\_child\_weight': 7}, mean: 0.80028, std: 0.03511, params: {'max\_depth': 6, 'min\_child\_weight': 1}, mean: 0.79843, std: 0.04511, params: {'max\_depth': 6, 'min\_child\_weight': 3}, mean: 0.79646, std: 0.04588, params: {'max\_depth': 6, 'min\_child\_weight': 5}, mean: 0.80076, std: 0.03813, params: {'max\_depth': 6, 'min\_child\_weight': 7}, mean: 0.78292, std: 0.02885, params: {'max\_depth': 8, 'min\_child\_weight': 1}, mean: 0.78969, std: 0.04313, params: {'max\_depth': 8, 'min\_child\_weight': 3}, mean: 0.79264, std: 0.03635, params: {'max\_depth': 8, 'min\_child\_weight': 5}, mean: 0.80155, std: 0.04231, params: {'max\_depth': 8, 'min\_child\_weight': 7}] {'max\_depth': 2, 'min\_child\_weight': 5} 0.809213481575

[mean: 0.79156, std: 0.04603, params: {'max\_depth': 1, 'min\_child\_weight': 4}, mean: 0.79118, std: 0.04772, params: {'max\_depth': 1, 'min\_child\_weight': 5}, mean: 0.79118, std: 0.04772, params: {'max\_depth': 1, 'min\_child\_weight': 6}, mean: 0.80817, std: 0.04350, params: {'max\_depth': 2, 'min\_child\_weight': 4}, mean: 0.80921, std: 0.04222, params: {'max\_depth': 2, 'min\_child\_weight': 5}, mean: 0.80630, std: 0.04281, params: {'max\_depth': 2, 'min\_child\_weight': 6}, mean: 0.80880, std: 0.04994, params: {'max\_depth': 3, 'min\_child\_weight': 4}, mean: 0.80349, std: 0.03971, params: {'max\_depth': 3, 'min\_child\_weight': 5}, mean: 0.80639, std: 0.03829, params: {'max\_depth': 3, 'min\_child\_weight': 6}] {'max\_depth': 2, 'min\_child\_weight': 5} 0.809213481575

[mean: 0.80921, std: 0.04222, params: {'gamma': 0.0}, mean: 0.80920, std: 0.04217, params: {'gamma': 0.1}, mean: 0.80819, std: 0.04329, params: {'gamma': 0.2}, mean: 0.80801, std: 0.04327, params: {'gamma': 0.3}, mean: 0.81002, std: 0.04381, params: {'gamma': 0.4}] {'gamma': 0.4} 0.810022413659

[mean: 0.80234, std: 0.04057, params: {'colsample\_bytree': 0.6, 'subsample': 0.6}, mean: 0.80343, std: 0.04276, params: {'colsample\_bytree': 0.6, 'subsample': 0.7}, mean: 0.80399, std: 0.04131, params: {'colsample\_bytree': 0.6, 'subsample': 0.8}, mean: 0.80335, std: 0.04099, params: {'colsample\_bytree': 0.6, 'subsample': 0.9}, mean: 0.80605, std: 0.04321, params: {'colsample\_bytree': 0.7, 'subsample': 0.6}, mean: 0.80830, std: 0.04147, params: {'colsample\_bytree': 0.7, 'subsample': 0.7}, mean: 0.80688, std: 0.04410, params: {'colsample\_bytree': 0.7, 'subsample': 0.8}, mean: 0.79923, std: 0.04012, params: {'colsample\_bytree': 0.7, 'subsample': 0.9}, mean: 0.80349, std: 0.04222, params: {'colsample\_bytree': 0.8, 'subsample': 0.6}, mean: 0.80599, std: 0.04047, params: {'colsample\_bytree': 0.8, 'subsample': 0.7}, mean: 0.81002, std: 0.04381, params: {'colsample\_bytree': 0.8, 'subsample': 0.8}, mean: 0.80428, std: 0.04121, params: {'colsample\_bytree': 0.8, 'subsample': 0.9}, mean: 0.80532, std: 0.04858, params: {'colsample\_bytree': 0.9, 'subsample': 0.6}, mean: 0.80680, std: 0.04079, params: {'colsample\_bytree': 0.9, 'subsample': 0.7}, mean: 0.80626, std: 0.03974, params: {'colsample\_bytree': 0.9, 'subsample': 0.8}, mean: 0.80462, std: 0.04318, params: {'colsample\_bytree': 0.9, 'subsample': 0.9}] {'colsample\_bytree': 0.8, 'subsample': 0.8} 0.810022413659

[Finished in 317.4s]

[mean: 0.80448, std: 0.04431, params: {'reg\_lambda': 0}, mean: 0.80627, std: 0.04480, params: {'reg\_lambda': 0.1}, mean: 0.81002, std: 0.04381, params: {'reg\_lambda': 1}, mean: 0.80649, std: 0.04235, params: {'reg\_lambda': 10}, mean: 0.80361, std: 0.04732, params: {'reg\_lambda': 100}] {'reg\_lambda': 1} 0.810022413659

[mean: 0.80627, std: 0.04455, params: {'reg\_lambda': 0.5}, mean: 0.81002, std: 0.04381, params: {'reg\_lambda': 1}, mean: 0.80869, std: 0.04132, params: {'reg\_lambda': 2}, mean: 0.80647, std: 0.04280, params: {'reg\_lambda': 5}] {'reg\_lambda': 1} 0.810022413659

[mean: 0.81002, std: 0.04381, params: {'max\_delta\_step': 1}, mean: 0.80667, std: 0.04034, params: {'max\_delta\_step': 3}, mean: 0.80667, std: 0.04034, params: {'max\_delta\_step': 5}, mean: 0.80667, std: 0.04034, params: {'max\_delta\_step': 7}, mean: 0.80667, std: 0.04034, params: {'max\_delta\_step': 9}] {'max\_delta\_step': 1} 0.810022413659