Data Description-

Santander's competition is about predicting which clients are going to make a transfer in the future. The available dataset is split between train and test, with 200,000 clients in each . There are 200 variables available (0, 199), but no information about them is given other than the numbers they hold.

we are asked to predict if a customer will make a transaction or not regardless of the amount of money transacted. Hence it is binary classification problem. In the data,the features given are numeric and anonymized. 0 indicate person doesn't make the transfer. 1 denotes person make the transfer.

Performance metric used-

1-AUC

Loading of dataset

In [0]:

```
#importing libraries
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import StratifiedKFold,GridSearchCV,RandomizedSearchCV
import lightgbm as lgb
```

In [0]:

```
#load the data
train_df=pd.read_csv("train.csv")
test_df=pd.read_csv("test.csv")
```

In [9]:

```
#columns in data
print("columns are:-",train_df.columns)
```

```
columns are:- Index(['ID_code', 'target', 'var_0', 'var_1', 'var_2', 'var_3', 'var_4',
       'var_5', 'var_6', 'var_7',
       ...
       'var_190', 'var_191', 'var_192', 'var_193', 'var_194', 'var_195',
       'var_196', 'var_197', 'var_198', 'var_199'],
      dtype='object', length=202)
```

In [10]:

```
#shape of data
print("train shape-",train_df.shape)
print("test shape",test_df.shape)
```

```
train shape- (200000, 202)
test shape (200000, 201)
```

In [0]:

```
#head
train_df.head()
```

Out[0]:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	var_10	var_11	var_
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	-4.9200	5.7470	2.9252	3.1821	14.01
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	3.1468	8.0851	-0.4032	8.0585	14.02
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193	5.9525	-0.3249	-11.2648	14.19
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.8609	8.2450	2.3061	2.8102	13.84
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.2654	7.6784	-9.4458	-12.1419	13.84

5 rows × 202 columns

In [0]:

```
test_df.head()
```

Out[0]:

	ID_code	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	var_10	var_11	var_12	va
0	test_0	11.0656	7.7798	12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	8.8100	-2.0248	-4.3554	13.9696	0.3
1	test_1	8.5304	1.2543	11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	5.9739	-1.3809	-0.3310	14.1129	2.5
2	test_2	5.4827	-10.3581	10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	8.3442	-4.7057	-3.0422	13.6751	3.8
3	test_3	8.5374	-1.3222	12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	7.4578	0.0095	-5.0659	14.0526	13.
4	test_4	11.7058	-0.1327	14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	7.1437	5.1025	-3.2827	14.1013	8.9

5 rows × 201 columns

EDA

In [0]:

```
#description of data
train_df.describe()
```

Out[0]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	20000
mean	0.100490	10.679914	-1.627622	10.715192	6.796529	11.078333	-5.065317	5.4089
std	0.300653	3.040051	4.050044	2.640894	2.043319	1.623150	7.863267	0.8666
min	0.000000	0.408400	-15.043400	2.117100	-0.040200	5.074800	-32.562600	2.3473
25%	0.000000	8.453850	-4.740025	8.722475	5.254075	9.883175	-11.200350	4.7677
50%	0.000000	10.524750	-1.608050	10.580000	6.825000	11.108250	-4.833150	5.3851

	target	var_0	var_1	var_2	var_3	var_4	var_5	
75%	0.000000	12.758200	1.358625	12.516700	8.324100	12.261125	0.924800	6.0030
max	1.000000	20.315000	10.376800	19.353000	13.188300	16.671400	17.251600	8.4477

8 rows × 201 columns

In [0]:

```
#description of data
test_df.describe()
```

Out[0]:

	var_0	var_1	var_2	var_3	var_4	var_5	var_6	
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	20000
mean	10.658737	-1.624244	10.707452	6.788214	11.076399	-5.050558	5.415164	16.529
std	3.036716	4.040509	2.633888	2.052724	1.616456	7.869293	0.864686	3.4244
min	0.188700	-15.043400	2.355200	-0.022400	5.484400	-27.767000	2.216400	5.7137
25%	8.442975	-4.700125	8.735600	5.230500	9.891075	-11.201400	4.772600	13.933
50%	10.513800	-1.590500	10.560700	6.822350	11.099750	-4.834100	5.391600	16.422
75%	12.739600	1.343400	12.495025	8.327600	12.253400	0.942575	6.005800	19.094
max	22.323400	9.385100	18.714100	13.142000	16.037100	17.253700	8.302500	28.292

8 rows × 200 columns

mean values are distributed over the large range.

---

std dev for some features are high.

In [0]:

```
#checking for null entries in train
train_df.isnull().any().any()
```

Out[0]:

```
False
```

In [0]:

```
#checking for null entries in train
test_df.isnull().any().any()
```

Out[0]:

```
False
```

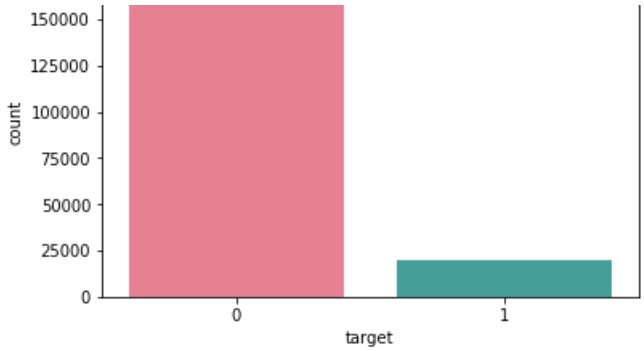
there is no missing value in train & test

In [0]:

```
#class imbalance
sns.countplot(x='target',data=train_df,palette="husl")
```

Out[0]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4126109b38>
```



In [0]:

```
train_df['target'].value_counts()
```

Out[0]:

```
0    179902
1     20098
Name: target, dtype: int64
```

In [0]:

```
print("percent of class 1 data points is", 100 * train_df["target"].value_counts()[1]/train_df.shape[0],'%')
print("percent of class 0 data points is", 100 * train_df["target"].value_counts()[0]/train_df.shape[0],'%')
```

```
percent of class 1 data points is 10.049 %
percent of class 0 data points is 89.951 %
```

highly imbalanced data

In [0]:

```
#number of unique value in each features in train
for col in train_df.columns[2:]:
    print("Number of unique values of {} : {}".format(col, train_df[col].nunique()))
```

```
Number of unique values of var_0 : 94672
Number of unique values of var_1 : 108932
Number of unique values of var_2 : 86555
Number of unique values of var_3 : 74597
Number of unique values of var_4 : 63515
Number of unique values of var_5 : 141029
Number of unique values of var_6 : 38599
Number of unique values of var_7 : 103063
Number of unique values of var_8 : 98617
Number of unique values of var_9 : 49417
Number of unique values of var_10 : 128764
Number of unique values of var_11 : 130193
Number of unique values of var_12 : 9561
Number of unique values of var_13 : 115181
Number of unique values of var_14 : 79122
Number of unique values of var_15 : 19810
Number of unique values of var_16 : 86918
Number of unique values of var_17 : 137823
Number of unique values of var_18 : 139515
Number of unique values of var_19 : 144180
Number of unique values of var_20 : 127764
Number of unique values of var_21 : 140062
Number of unique values of var_22 : 90660
Number of unique values of var_23 : 24913
Number of unique values of var_24 : 105101
Number of unique values of var_25 : 14853
Number of unique values of var_26 : 127089
Number of unique values of var_27 : 60185
Number of unique values of var_28 : 35859
Number of unique values of var_29 : 88339
```

```
Number of unique values of var_30 : 145977
Number of unique values of var_31 : 77388
Number of unique values of var_32 : 85964
Number of unique values of var_33 : 112239
Number of unique values of var_34 : 25164
Number of unique values of var_35 : 122384
Number of unique values of var_36 : 96404
Number of unique values of var_37 : 79040
Number of unique values of var_38 : 115366
Number of unique values of var_39 : 112674
Number of unique values of var_40 : 141878
Number of unique values of var_41 : 131896
Number of unique values of var_42 : 31592
Number of unique values of var_43 : 15188
Number of unique values of var_44 : 127702
Number of unique values of var_45 : 169968
Number of unique values of var_46 : 93450
Number of unique values of var_47 : 154781
Number of unique values of var_48 : 152039
Number of unique values of var_49 : 140641
Number of unique values of var_50 : 32308
Number of unique values of var_51 : 143455
Number of unique values of var_52 : 121313
Number of unique values of var_53 : 33460
Number of unique values of var_54 : 144776
Number of unique values of var_55 : 128077
Number of unique values of var_56 : 103045
Number of unique values of var_57 : 35545
Number of unique values of var_58 : 113907
Number of unique values of var_59 : 37744
Number of unique values of var_60 : 113763
Number of unique values of var_61 : 159369
Number of unique values of var_62 : 74777
Number of unique values of var_63 : 97098
Number of unique values of var_64 : 59379
Number of unique values of var_65 : 108347
Number of unique values of var_66 : 47722
Number of unique values of var_67 : 137253
Number of unique values of var_68 : 451
Number of unique values of var_69 : 110346
Number of unique values of var_70 : 153193
Number of unique values of var_71 : 13527
Number of unique values of var_72 : 110114
Number of unique values of var_73 : 142582
Number of unique values of var_74 : 161058
Number of unique values of var_75 : 129383
Number of unique values of var_76 : 139317
Number of unique values of var_77 : 106809
Number of unique values of var_78 : 72254
Number of unique values of var_79 : 53212
Number of unique values of var_80 : 136432
Number of unique values of var_81 : 79065
Number of unique values of var_82 : 144829
Number of unique values of var_83 : 144281
Number of unique values of var_84 : 133766
Number of unique values of var_85 : 108437
Number of unique values of var_86 : 140594
Number of unique values of var_87 : 125296
Number of unique values of var_88 : 84918
Number of unique values of var_89 : 103522
Number of unique values of var_90 : 157210
Number of unique values of var_91 : 7962
Number of unique values of var_92 : 110743
Number of unique values of var_93 : 26708
Number of unique values of var_94 : 89146
Number of unique values of var_95 : 29387
Number of unique values of var_96 : 148099
Number of unique values of var_97 : 158739
Number of unique values of var_98 : 33266
Number of unique values of var_99 : 69300
Number of unique values of var_100 : 150727
Number of unique values of var_101 : 122295
Number of unique values of var_102 : 146237
Number of unique values of var_103 : 9376
Number of unique values of var_104 : 72627
Number of unique values of var_105 : 39115
Number of unique values of var_106 : 71065
```

```
Number of unique values of var_107 : 137827
Number of unique values of var_108 : 8525
Number of unique values of var_109 : 112172
Number of unique values of var_110 : 106121
Number of unique values of var_111 : 46464
Number of unique values of var_112 : 60482
Number of unique values of var_113 : 116496
Number of unique values of var_114 : 43084
Number of unique values of var_115 : 86729
Number of unique values of var_116 : 63467
Number of unique values of var_117 : 164469
Number of unique values of var_118 : 143667
Number of unique values of var_119 : 112403
Number of unique values of var_120 : 158269
Number of unique values of var_121 : 64695
Number of unique values of var_122 : 121767
Number of unique values of var_123 : 129893
Number of unique values of var_124 : 91022
Number of unique values of var_125 : 16059
Number of unique values of var_126 : 32411
Number of unique values of var_127 : 95710
Number of unique values of var_128 : 98200
Number of unique values of var_129 : 113425
Number of unique values of var_130 : 36638
Number of unique values of var_131 : 21464
Number of unique values of var_132 : 57923
Number of unique values of var_133 : 19236
Number of unique values of var_134 : 131619
Number of unique values of var_135 : 140774
Number of unique values of var_136 : 156615
Number of unique values of var_137 : 144397
Number of unique values of var_138 : 117428
Number of unique values of var_139 : 137294
Number of unique values of var_140 : 121384
Number of unique values of var_141 : 134443
Number of unique values of var_142 : 128613
Number of unique values of var_143 : 94372
Number of unique values of var_144 : 40595
Number of unique values of var_145 : 108526
Number of unique values of var_146 : 84314
Number of unique values of var_147 : 137559
Number of unique values of var_148 : 10608
Number of unique values of var_149 : 148504
Number of unique values of var_150 : 83660
Number of unique values of var_151 : 109667
Number of unique values of var_152 : 95823
Number of unique values of var_153 : 73728
Number of unique values of var_154 : 119342
Number of unique values of var_155 : 127457
Number of unique values of var_156 : 40634
Number of unique values of var_157 : 126534
Number of unique values of var_158 : 144556
Number of unique values of var_159 : 112830
Number of unique values of var_160 : 156274
Number of unique values of var_161 : 11071
Number of unique values of var_162 : 57396
Number of unique values of var_163 : 123168
Number of unique values of var_164 : 122744
Number of unique values of var_165 : 119403
Number of unique values of var_166 : 17902
Number of unique values of var_167 : 140954
Number of unique values of var_168 : 97227
Number of unique values of var_169 : 18242
Number of unique values of var_170 : 113720
Number of unique values of var_171 : 125914
Number of unique values of var_172 : 143366
Number of unique values of var_173 : 128120
Number of unique values of var_174 : 134945
Number of unique values of var_175 : 92659
Number of unique values of var_176 : 142521
Number of unique values of var_177 : 85720
Number of unique values of var_178 : 145235
Number of unique values of var_179 : 90090
Number of unique values of var_180 : 123477
Number of unique values of var_181 : 56164
Number of unique values of var_182 : 149195
Number of unique values of var_183 : 117529
```

```
Number of unique values of var_184 : 145184
Number of unique values of var_185 : 120747
Number of unique values of var_186 : 98060
Number of unique values of var_187 : 157031
Number of unique values of var_188 : 108813
Number of unique values of var_189 : 41764
Number of unique values of var_190 : 114959
Number of unique values of var_191 : 94266
Number of unique values of var_192 : 59065
Number of unique values of var_193 : 110557
Number of unique values of var_194 : 97069
Number of unique values of var_195 : 57870
Number of unique values of var_196 : 125560
Number of unique values of var_197 : 40537
Number of unique values of var_198 : 94153
Number of unique values of var_199 : 149430
```

Most features have more than thousands of values for each variable except var_68. so var_68 may be categorical feature.

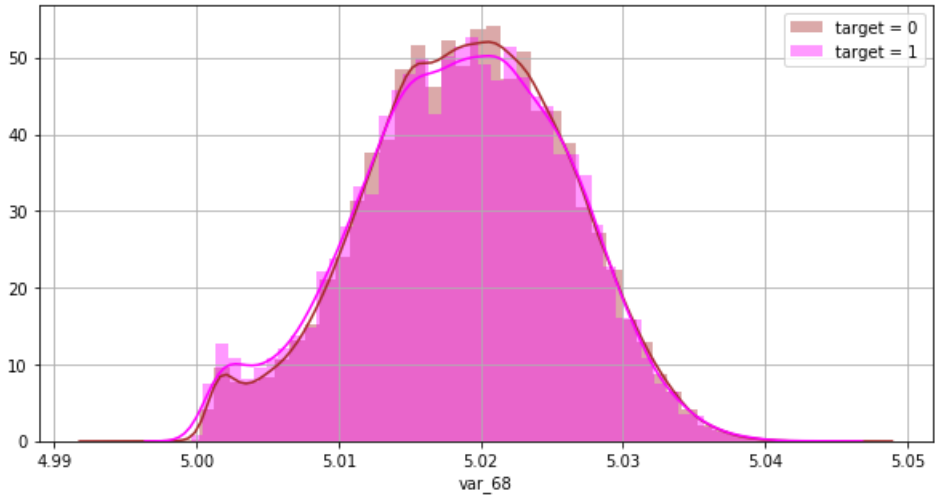
let's see the distribution of var_68

In [0]:

```
#seperating target 0 & target 1 data
t0=train_df[train_df['target']==0]
t1=train_df[train_df['target']==1]
```

In [0]:

```
#distribution of var_68
plt.figure(figsize=(10,5))
sns.distplot(t0['var_68'],color="brown", label='target = 0')
sns.distplot(t1['var_68'],color="magenta",label='target = 1')
plt.legend()
plt.grid()
plt.show()
```

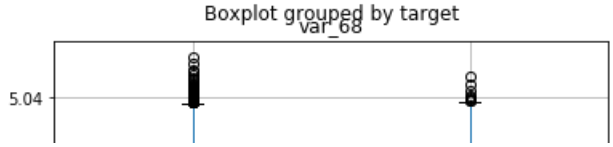


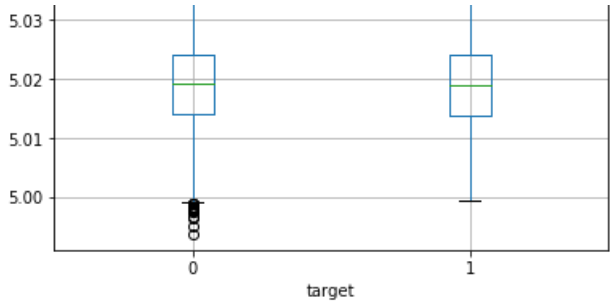
In [0]:

```
#violin plot
train_df.boxplot('var_68',by='target')
```

Out[0]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f00cad6dc18>
```





dist. is mostly overlapping for both the class.

In [11]:

```
#number of unique value in each features in test
for col in test_df.columns[2:]:
    print("Number of unique values of {} : {}".format(col, test_df[col].nunique()))
```

```
Number of unique values of var_1 : 71661
Number of unique values of var_2 : 61865
Number of unique values of var_3 : 56507
Number of unique values of var_4 : 49995
Number of unique values of var_5 : 83228
Number of unique values of var_6 : 33273
Number of unique values of var_7 : 69487
Number of unique values of var_8 : 67521
Number of unique values of var_9 : 41583
Number of unique values of var_10 : 79221
Number of unique values of var_11 : 79749
Number of unique values of var_12 : 9121
Number of unique values of var_13 : 74037
Number of unique values of var_14 : 58951
Number of unique values of var_15 : 18253
Number of unique values of var_16 : 61906
Number of unique values of var_17 : 82518
Number of unique values of var_18 : 82682
Number of unique values of var_19 : 84370
Number of unique values of var_20 : 78645
Number of unique values of var_21 : 82738
Number of unique values of var_22 : 63855
Number of unique values of var_23 : 22619
Number of unique values of var_24 : 70202
Number of unique values of var_25 : 13728
Number of unique values of var_26 : 78260
Number of unique values of var_27 : 48428
Number of unique values of var_28 : 31321
Number of unique values of var_29 : 62618
Number of unique values of var_30 : 84985
Number of unique values of var_31 : 57146
Number of unique values of var_32 : 61890
Number of unique values of var_33 : 73157
Number of unique values of var_34 : 22954
Number of unique values of var_35 : 76756
Number of unique values of var_36 : 66309
Number of unique values of var_37 : 58742
Number of unique values of var_38 : 74294
Number of unique values of var_39 : 73292
Number of unique values of var_40 : 83405
Number of unique values of var_41 : 80327
Number of unique values of var_42 : 28163
Number of unique values of var_43 : 14288
Number of unique values of var_44 : 78457
Number of unique values of var_45 : 92058
Number of unique values of var_46 : 65189
Number of unique values of var_47 : 87427
Number of unique values of var_48 : 86929
Number of unique values of var_49 : 82973
Number of unique values of var_50 : 28412
Number of unique values of var_51 : 83881
Number of unique values of var_52 : 76266
Number of unique values of var_53 : 29631
Number of unique values of var_54 : 84548
Number of unique values of var_55 : 79114
```

```
Number of unique values of var_55 : 79114
Number of unique values of var_56 : 69316
Number of unique values of var_57 : 31286
Number of unique values of var_58 : 73482
Number of unique values of var_59 : 32888
Number of unique values of var_60 : 73575
Number of unique values of var_61 : 88874
Number of unique values of var_62 : 55891
Number of unique values of var_63 : 66913
Number of unique values of var_64 : 47632
Number of unique values of var_65 : 71421
Number of unique values of var_66 : 40071
Number of unique values of var_67 : 81931
Number of unique values of var_68 : 428
Number of unique values of var_69 : 72217
Number of unique values of var_70 : 86863
Number of unique values of var_71 : 12604
Number of unique values of var_72 : 72162
Number of unique values of var_73 : 83685
Number of unique values of var_74 : 89343
Number of unique values of var_75 : 79183
Number of unique values of var_76 : 82703
Number of unique values of var_77 : 71025
Number of unique values of var_78 : 54942
Number of unique values of var_79 : 44041
Number of unique values of var_80 : 81543
Number of unique values of var_81 : 58200
Number of unique values of var_82 : 84426
Number of unique values of var_83 : 84275
Number of unique values of var_84 : 80933
Number of unique values of var_85 : 71490
Number of unique values of var_86 : 82894
Number of unique values of var_87 : 77815
Number of unique values of var_88 : 61261
Number of unique values of var_89 : 69297
Number of unique values of var_90 : 88329
Number of unique values of var_91 : 7569
Number of unique values of var_92 : 72603
Number of unique values of var_93 : 23637
Number of unique values of var_94 : 63414
Number of unique values of var_95 : 26068
Number of unique values of var_96 : 85505
Number of unique values of var_97 : 88491
Number of unique values of var_98 : 29142
Number of unique values of var_99 : 53278
Number of unique values of var_100 : 86300
Number of unique values of var_101 : 76802
Number of unique values of var_102 : 84953
Number of unique values of var_103 : 8828
Number of unique values of var_104 : 55066
Number of unique values of var_105 : 33485
Number of unique values of var_106 : 54037
Number of unique values of var_107 : 82101
Number of unique values of var_108 : 8188
Number of unique values of var_109 : 73053
Number of unique values of var_110 : 70177
Number of unique values of var_111 : 38860
Number of unique values of var_112 : 48537
Number of unique values of var_113 : 74676
Number of unique values of var_114 : 36606
Number of unique values of var_115 : 61894
Number of unique values of var_116 : 50209
Number of unique values of var_117 : 90342
Number of unique values of var_118 : 84091
Number of unique values of var_119 : 72992
Number of unique values of var_120 : 88478
Number of unique values of var_121 : 50805
Number of unique values of var_122 : 76235
Number of unique values of var_123 : 79307
Number of unique values of var_124 : 63864
Number of unique values of var_125 : 14744
Number of unique values of var_126 : 29224
Number of unique values of var_127 : 66401
Number of unique values of var_128 : 67606
Number of unique values of var_129 : 73434
Number of unique values of var_130 : 32095
Number of unique values of var_131 : 19765
Number of unique values of var_132 : 46669
```

```
Number of unique values of var_132 : 46669
Number of unique values of var_133 : 17411
Number of unique values of var_134 : 80100
Number of unique values of var_135 : 83073
Number of unique values of var_136 : 88160
Number of unique values of var_137 : 84520
Number of unique values of var_138 : 75048
Number of unique values of var_139 : 81961
Number of unique values of var_140 : 76537
Number of unique values of var_141 : 81076
Number of unique values of var_142 : 79095
Number of unique values of var_143 : 65590
Number of unique values of var_144 : 35056
Number of unique values of var_145 : 71625
Number of unique values of var_146 : 60676
Number of unique values of var_147 : 82134
Number of unique values of var_148 : 9964
Number of unique values of var_149 : 85452
Number of unique values of var_150 : 60849
Number of unique values of var_151 : 72184
Number of unique values of var_152 : 66139
Number of unique values of var_153 : 55749
Number of unique values of var_154 : 75651
Number of unique values of var_155 : 78595
Number of unique values of var_156 : 35466
Number of unique values of var_157 : 78323
Number of unique values of var_158 : 84373
Number of unique values of var_159 : 73068
Number of unique values of var_160 : 88104
Number of unique values of var_161 : 10506
Number of unique values of var_162 : 46306
Number of unique values of var_163 : 76976
Number of unique values of var_164 : 77121
Number of unique values of var_165 : 75703
Number of unique values of var_166 : 16683
Number of unique values of var_167 : 83182
Number of unique values of var_168 : 66656
Number of unique values of var_169 : 16759
Number of unique values of var_170 : 73455
Number of unique values of var_171 : 77918
Number of unique values of var_172 : 83916
Number of unique values of var_173 : 78845
Number of unique values of var_174 : 80987
Number of unique values of var_175 : 64937
Number of unique values of var_176 : 83905
Number of unique values of var_177 : 61780
Number of unique values of var_178 : 84537
Number of unique values of var_179 : 63473
Number of unique values of var_180 : 77195
Number of unique values of var_181 : 45307
Number of unique values of var_182 : 85984
Number of unique values of var_183 : 75174
Number of unique values of var_184 : 84500
Number of unique values of var_185 : 76106
Number of unique values of var_186 : 67204
Number of unique values of var_187 : 88290
Number of unique values of var_188 : 71322
Number of unique values of var_189 : 36220
Number of unique values of var_190 : 73782
Number of unique values of var_191 : 65460
Number of unique values of var_192 : 47041
Number of unique values of var_193 : 72171
Number of unique values of var_194 : 66942
Number of unique values of var_195 : 46482
Number of unique values of var_196 : 78038
Number of unique values of var_197 : 34817
Number of unique values of var_198 : 65262
Number of unique values of var_199 : 85933
```

here also var_68 has less unique value.

In [12]:

```
#checking correlation
cor=train_df.corr()
cor.head()
```

Out[12]:

	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	var_
target	1.000000	0.052390	0.050343	0.055870	0.011055	0.010915	0.030979	0.066731	-0.003025	0.019584	-0.042805	-0.0022
var_0	0.052390	1.000000	-0.000544	0.006573	0.003801	0.001326	0.003046	0.006983	0.002429	0.004962	-0.002613	0.0003
var_1	0.050343	-0.000544	1.000000	0.003980	0.000010	0.000303	-0.000902	0.003258	0.001511	0.004098	-0.000832	0.0028
var_2	0.055870	0.006573	0.003980	1.000000	0.001001	0.000723	0.001569	0.000883	-0.000991	0.002648	-0.001932	-0.0004
var_3	0.011055	0.003801	0.000010	0.001001	1.000000	-0.000322	0.003253	-0.000774	0.002500	0.003553	-0.000826	-0.0008

5 rows × 201 columns

In [0]:

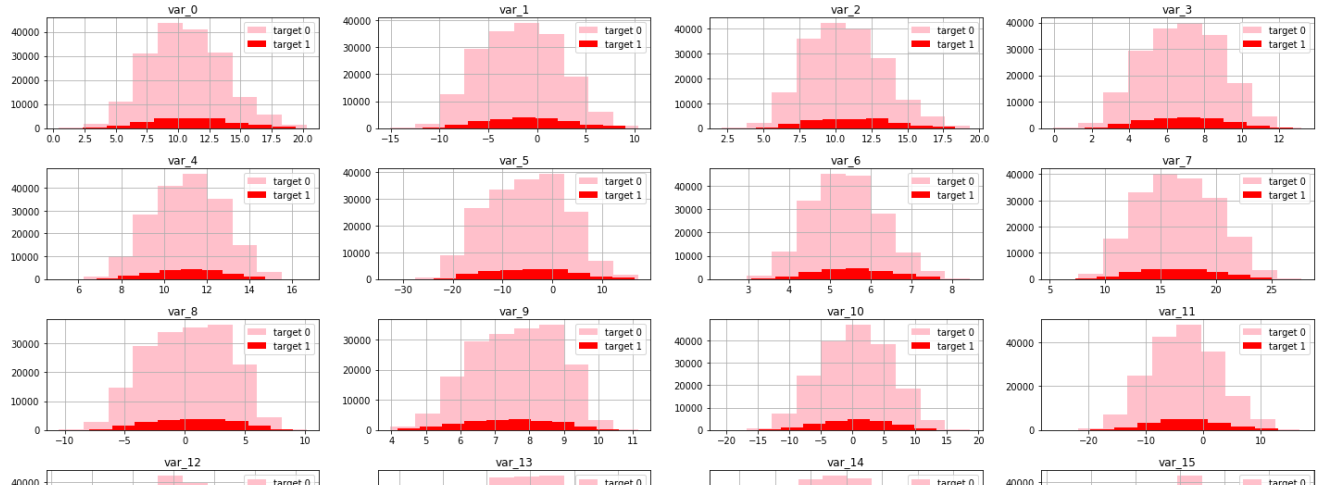
```
corr=abs(cor['target']).sort_values(ascending=False)
print('top 10 correlated features wrt target-:', corr[:10])
```

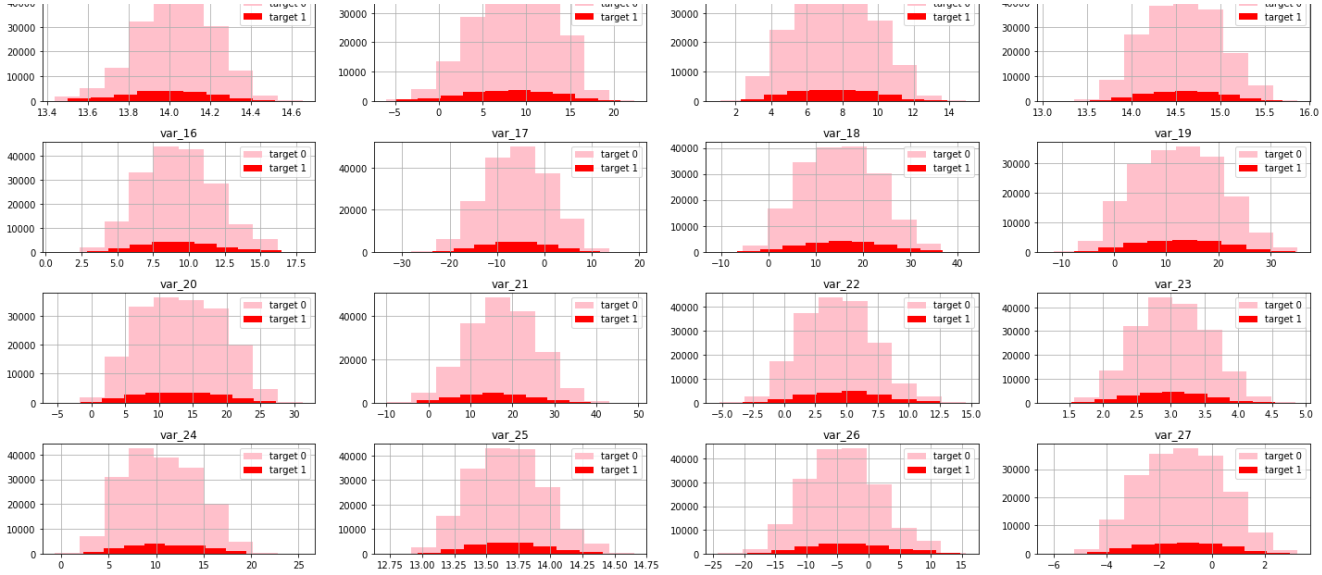
```
top 10 correlated features wrt target-: target     1.000000
var_81      0.080917
var_139     0.074080
var_12      0.069489
var_6       0.066731
var_110     0.064275
var_146     0.063644
var_53      0.063399
var_26      0.062422
var_76      0.061917
Name: target, dtype: float64
```

In [0]:

```
print('Distributions of 1st 28 features')
plt.figure(figsize=(20,16))
for i, col in enumerate(list(train_df.columns)[2:30]):
    plt.subplot(7,4,i + 1)
    plt.hist(t0[col],label='target 0',color='pink')
    plt.hist(t1[col],label='target 1',color='r')
    plt.title(col)
    plt.grid()
    plt.legend(loc='upper right')
    plt.tight_layout()
```

```
Distributions of 1st 28 features
```





In [0]:

```
print('Distributions of next 28 features')
plt.figure(figsize=(20,16))
for i,col in enumerate(list(train_df.columns)[30:58]):
    plt.subplot(7, 4,i + 1)
    plt.hist(t0[col],label='target 0',color='pink')
    plt.hist(t1[col],label='target 1',color='r')
    plt.title(col)
    plt.grid()
    plt.legend(loc='upper right')
    plt.tight_layout()
```

```
Distributions of next 28 features
```

there is no significant difference in values between the "transaction done" and "transaction not done".

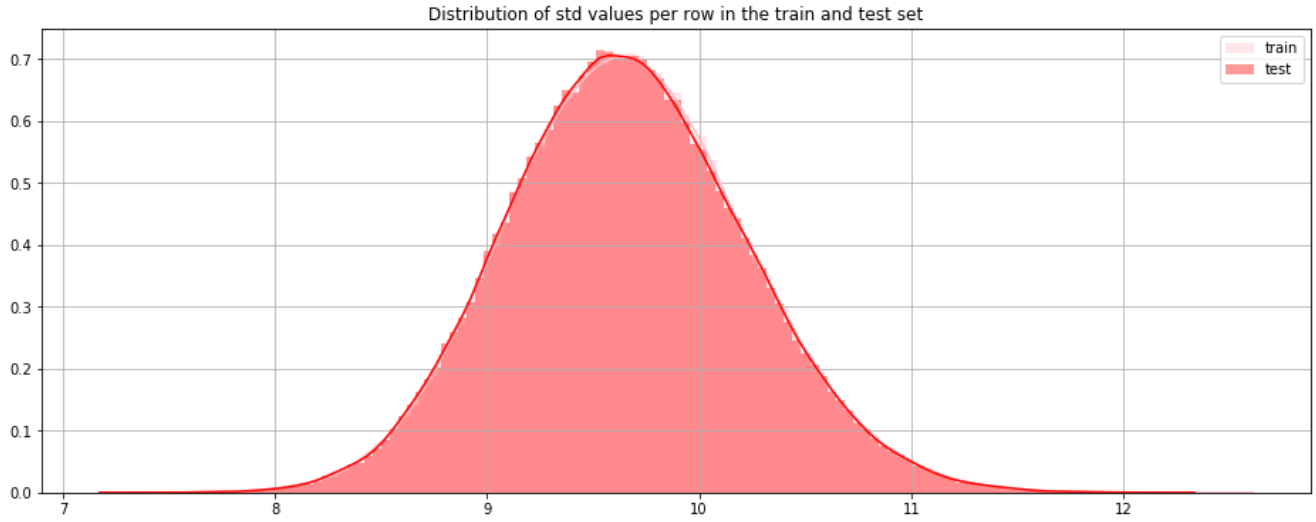
Distribution of some common fetaures in test & train data

In [0]:

```
plt.figure(figsize=(16,6))
feat= train_df.columns.values[2:202]
plt.title("Distribution of mean values per row in the train and test set")
sns.distplot(train_df[feat].mean(axis=1),color="yellow", kde=True,bins=120, label='train')
sns.distplot(test_df[feat].mean(axis=1),color="red", kde=True,bins=120, label='test')
plt.legend()
plt.grid()
plt.show()
```

In [0]:

```
plt.figure(figsize=(16,6))
plt.title("Distribution of std values per row in the train and test set")
sns.distplot(train_df[feat].std(axis=1),color="pink", kde=True,bins=120, label='train')
sns.distplot(test_df[feat].std(axis=1),color="red", kde=True,bins=120, label='test')
plt.legend()
plt.grid()
plt.show()
```



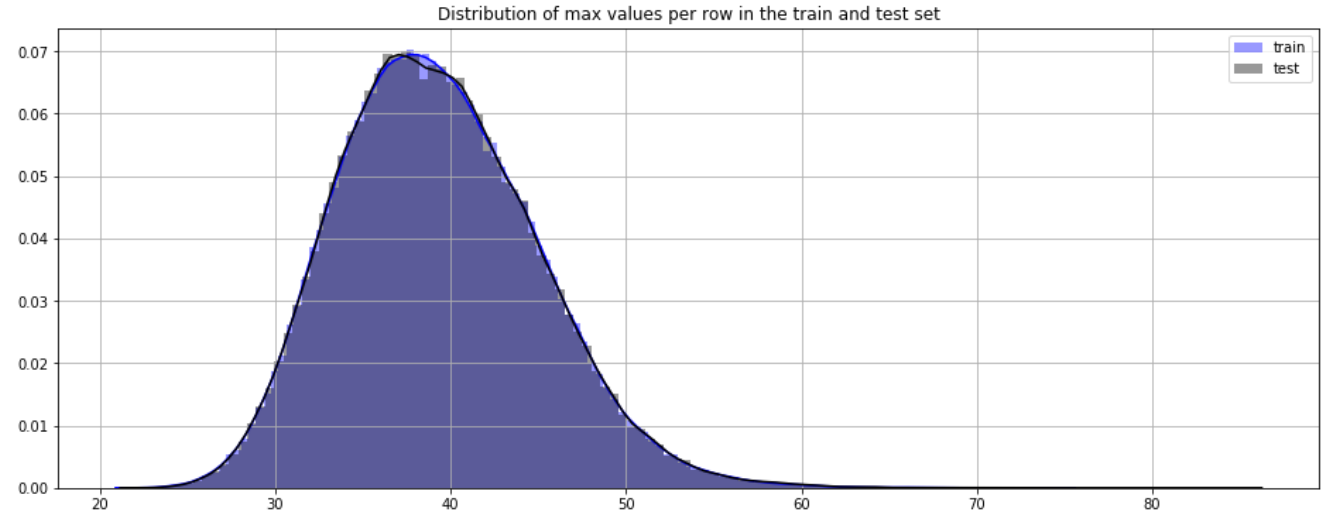
In [0]:

```
plt.figure(figsize=(16,6))
plt.title("Distribution of min values per row in the train and test set")
```

```
sns.distplot(train_df[features].min(axis=1),color="pink", kde=True,bins=120, label='train')
sns.distplot(test_df[features].min(axis=1),color="red", kde=True,bins=120, label='test')
plt.legend()
plt.grid()
plt.show()
```

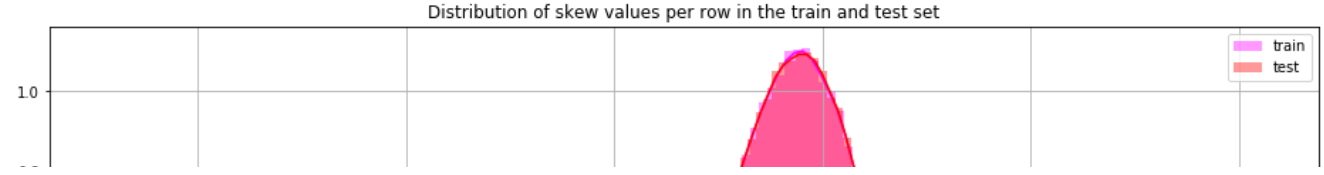
In [0]:

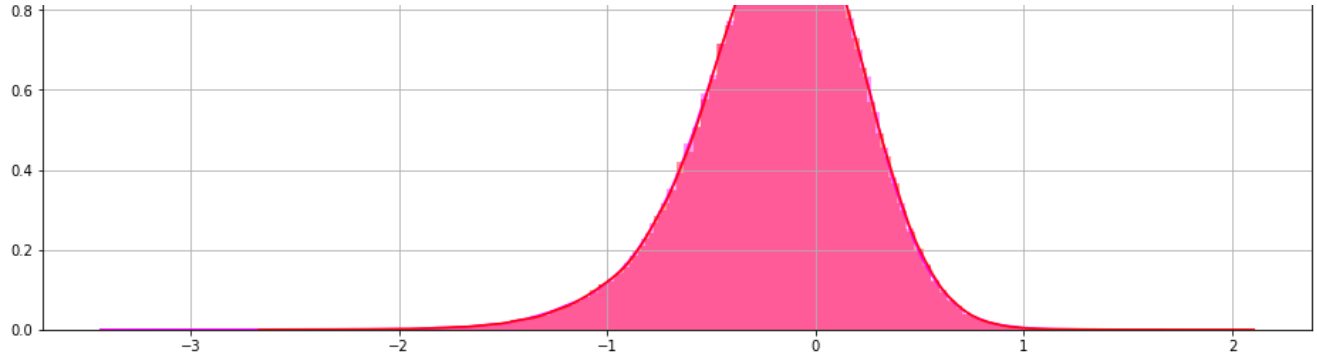
```
plt.figure(figsize=(16,6))
plt.title("Distribution of max values per row in the train and test set")
sns.distplot(train_df[feat].max(axis=1),color="b", kde=True,bins=120, label='train')
sns.distplot(test_df[feat].max(axis=1),color="black", kde=True,bins=120, label='test')
plt.legend()
plt.grid()
plt.show()
```



In [0]:

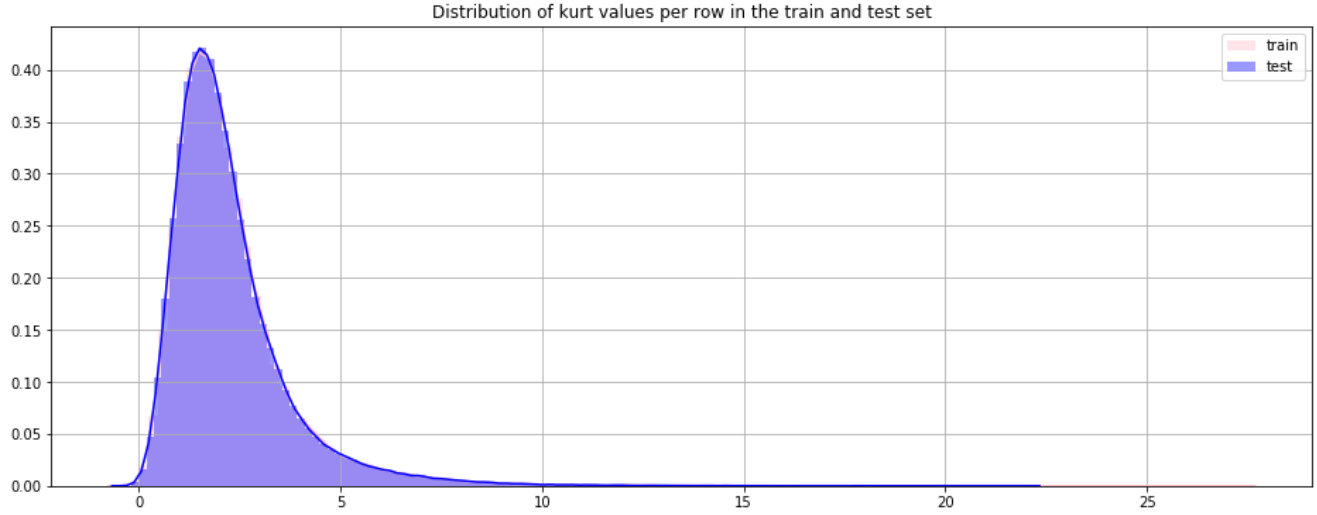
```
plt.figure(figsize=(16,6))
plt.title("Distribution of skew values per row in the train and test set")
sns.distplot(train_df[feat].skew(axis=1),color="magenta", kde=True,bins=120, label='train')
sns.distplot(test_df[feat].skew(axis=1),color="red", kde=True,bins=120, label='test')
plt.legend()
plt.grid()
plt.show()
```





In [0]:

```
plt.figure(figsize=(16,6))
plt.title("Distribution of kurt values per row in the train and test set")
sns.distplot(train_df[feat].kurt(axis=1),color="pink", kde=True,bins=120, label='train')
sns.distplot(test_df[feat].kurt(axis=1),color="blue", kde=True,bins=120, label='test')
plt.legend()
plt.grid()
plt.show()
```

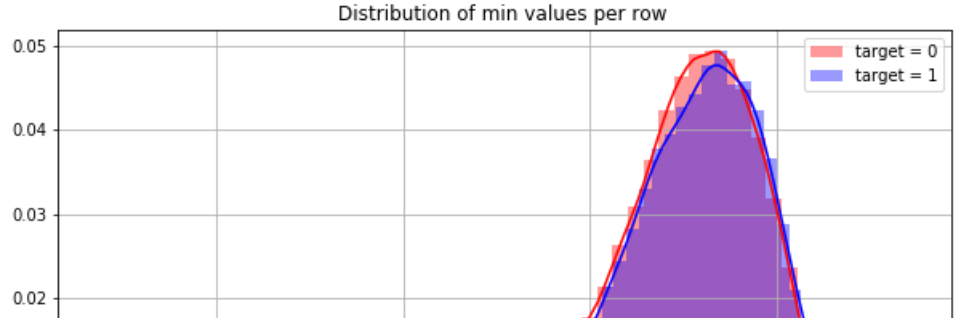


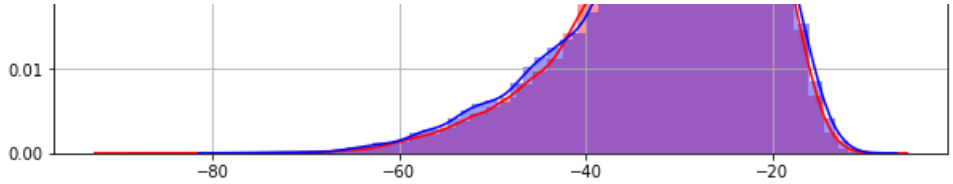
distribution is almost same for test and train data

visualization for adding some common features

In [0]:

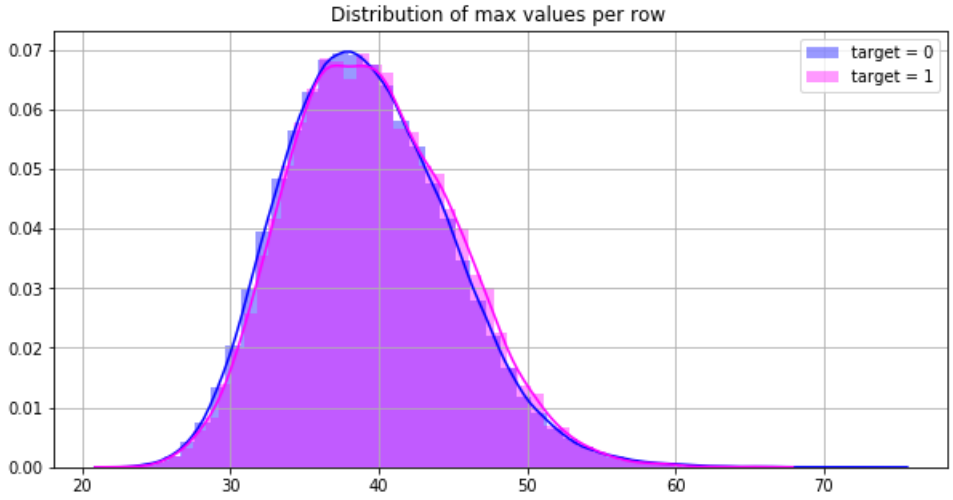
```
features =train_df.columns.values[2:202]
plt.figure(figsize=(10,5))
plt.title("Distribution of min values per row ")
sns.distplot(t0[features].min(axis=1),color="red", kde=True, label='target = 0')
sns.distplot(t1[features].min(axis=1),color="blue", kde=True, label='target = 1')
plt.legend()
plt.grid()
plt.show()
```





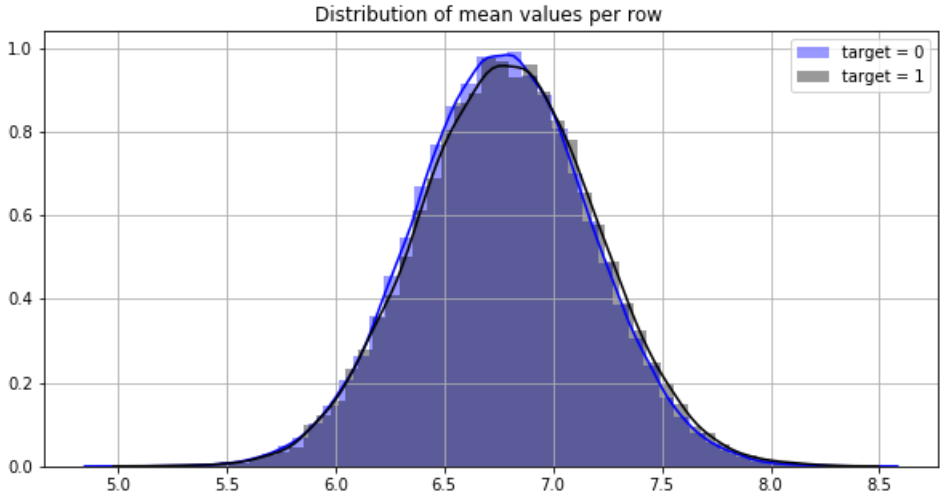
In [0]:

```
plt.figure(figsize=(10,5))
plt.title("Distribution of max values per row ")
sns.distplot(t0[features].max(axis=1),color="blue", kde=True, label='target = 0')
sns.distplot(t1[features].max(axis=1),color="magenta", kde=True, label='target = 1')
plt.legend()
plt.grid()
plt.show()
```



In [0]:

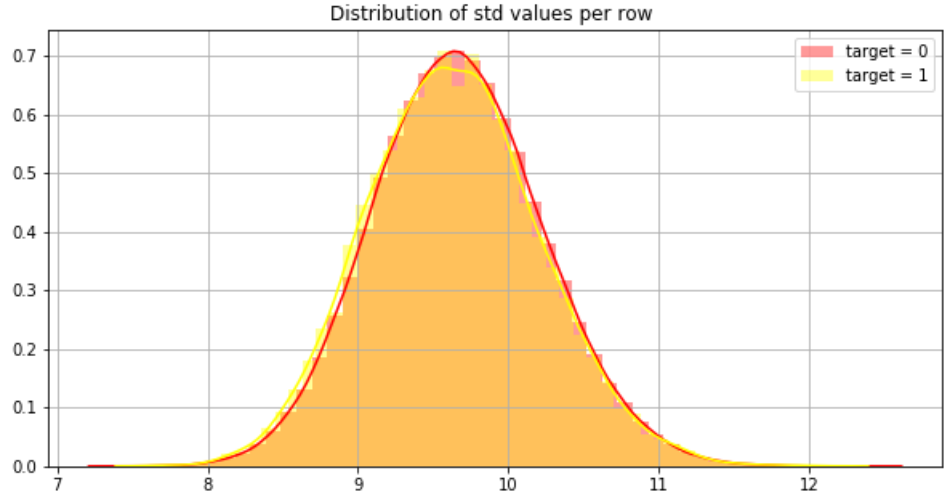
```
plt.figure(figsize=(10,5))
plt.title("Distribution of mean values per row ")
sns.distplot(t0[features].mean(axis=1),color="blue", kde=True, label='target = 0')
sns.distplot(t1[features].mean(axis=1),color="black", kde=True, label='target = 1')
plt.legend()
plt.grid()
plt.show()
```



In [0]:

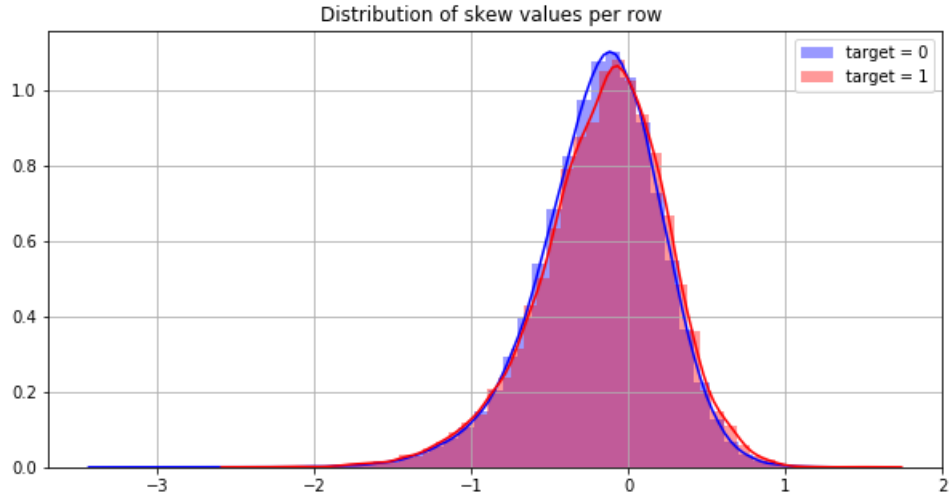
```
plt.figure(figsize=(10,5))
plt.title("Distribution of std values per row ")
sns.distplot(t0[features].std(axis=1),color="red", kde=True, label='target = 0')
sns.distplot(t1[features].std(axis=1),color="yellow", kde=True, label='target = 1')
```

```
plt.legend()
plt.grid()
plt.show()
```



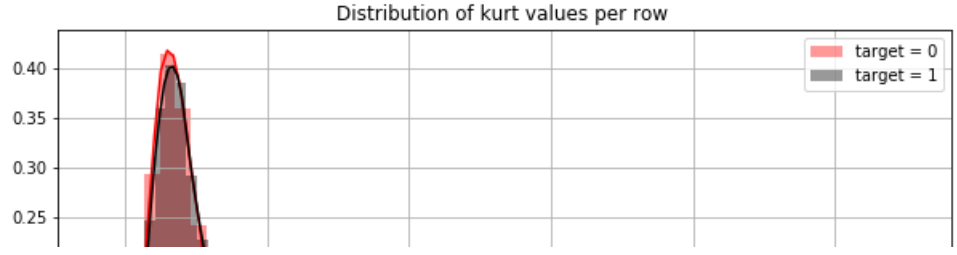
In [0]:

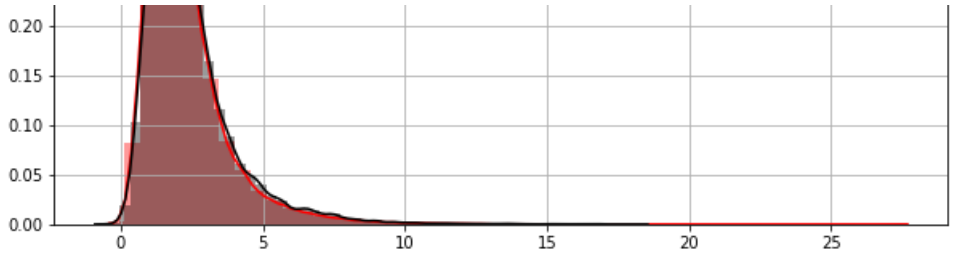
```
plt.figure(figsize=(10,5))
plt.title("Distribution of skew values per row ")
sns.distplot(t0[features].skew(axis=1),color="blue", kde=True, label='target = 0')
sns.distplot(t1[features].skew(axis=1),color="red", kde=True, label='target = 1')
plt.legend()
plt.grid()
plt.show()
```



In [0]:

```
plt.figure(figsize=(10,5))
plt.title("Distribution of kurt values per row ")
sns.distplot(t0[features].kurt(axis=1),color="red", kde=True, label='target = 0')
sns.distplot(t1[features].kurt(axis=1),color="black", kde=True, label='target = 1')
plt.legend()
plt.grid()
plt.show()
```





train data for target 0 & target 1 has almost similar distribution for mean,max,min,median,kurt & skew.

Feature engineering

In [0]:

```
#adding some common features
feat = train_df.columns.values[2:202]
for df in [test_df, train_df]:
    df['sum'] = df[feat].sum(axis=1)
    df['min'] = df[feat].min(axis=1)
    df['max'] = df[feat].max(axis=1)
    df['mean'] = df[feat].mean(axis=1)
    df['std'] = df[feat].std(axis=1)
    df['skew'] = df[feat].skew(axis=1)
    df['kurt'] = df[feat].kurtosis(axis=1)
    df['med'] = df[feat].median(axis=1)
```

In [0]:

```
print(train_df.shape)
print(test_df.shape)
```

```
(200000, 210)
(200000, 209)
```

preprocessing

In [0]:

```
feat = [col for col in train_df.columns if col not in ['ID_code', 'target']]
target = train_df['target']
```

In [0]:

```
x_test=test_df[feat]
x_train=train_df[feat]
```

In [0]:

```
#standarsiing the data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)

print(x_train.shape)
print(x_test.shape)
```

```
(200000, 208)
(200000, 208)
```

GBDT

GBDT

In [0]:

```
!pip install scikit-optimize
```

```
Collecting scikit-optimize
  Downloading
https://files.pythonhosted.org/packages/f4/44/60f82c97d1caa98752c7da2c1681cab5c7a390a0fdd3a55fac672
cac/scikit_optimize-0.5.2-py2.py3-none-any.whl (74kB)
    |████████████████████████████████| 81kB 6.8MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from scikit-
optimize) (1.16.4)
Requirement already satisfied: scikit-learn>=0.19.1 in /usr/local/lib/python3.6/dist-packages
(from scikit-optimize) (0.21.3)
Requirement already satisfied: scipy>=0.14.0 in /usr/local/lib/python3.6/dist-packages (from
scikit-optimize) (1.3.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from
scikit-learn>=0.19.1->scikit-optimize) (0.13.2)
Installing collected packages: scikit-optimize
Successfully installed scikit-optimize-0.5.2
```

In [0]:

```
import xgboost as xgb
from skopt import BayesSearchCV
from skopt.space import Real, Categorical, Integer
from sklearn.model_selection import train_test_split
```

In [0]:

```
#parameter tuning
#https://scikit-optimize.github.io/#skopt.BayesSearchCV
X_train, X_val, y_train, y_val = train_test_split(x_train, target, train_size=0.80, random_state=0)
opt = BayesSearchCV(
    xgb.XGBClassifier(metric='auc'),
    {
        'max_depth': (8,10),
        'alpha':(0.1,0.2),
        'gamma':(0.3,0.4),
        'learning_rate': (0.04, 0.05),
    },n_jobs=-1, cv=2)

opt.fit(X_train, y_train)
print("val. score: %s" % opt.best_score_)
print("test score: %s" % opt.score(X_val, y_val))
print("Best parameters: ", opt.best_params_)
```

it is taking lot of time more than 6 hrs on colab so tried gbdt without tuning hyperparameter

In [0]:

```
params = {'tree_method':'gpu_hist', 'max_depth':10, 'alpha':0.2,
          'gamma':0.4, 'subsample':0.5, 'scale_pos_weight':1, 'learning_rate': 0.05,
          'silent': 1, 'objective':'binary:logistic', 'eval_metric': 'auc',
          'n_gpus': 1}
```

In [0]:

```
#https://xgboost.readthedocs.io/en/latest/parameter.html
%%time

skf = StratifiedKFold(n_splits=8, shuffle=True, random_state=1234)
rem_samp = np.zeros(len(train_df))
pred = np.zeros(len(test_df))



for i,(tr_idx, val_idx) in enumerate(skf.split(x_train,target.values)):
    print("\nFold {}".format(i))
    xgb_train = xgb.DMatrix(x_train[tr_idx],target[tr_idx])
```

```
    xgb_val = xgb.DMatrix(x_train[val_idx],target[val_idx])

    clf = xgb.train(params, xgb_train,1000, evals=[(xgb_train, "train"), (xgb_val, "eval")],
                early_stopping_rounds=500, verbose_eval=False)

    rem_samp[val_idx] = clf.predict(xgb.DMatrix(x_train[val_idx]))

    pred += clf.predict(xgb.DMatrix(x_test)) / 8

print("CV AUC: {}".format(roc_auc_score(target,rem_samp)))
```

```
Fold 0

Fold 1

Fold 2

Fold 3

Fold 4

Fold 5

Fold 6

Fold 7
CV AUC: 0.8820962643686727
CPU times: user 23min 25s, sys: 11min 21s, total: 34min 46s
Wall time: 32min 12s
```

LGB

In [0]:

```
bay_tr_index, bay_val_index  = list(StratifiedKFold(n_splits=2, shuffle=True, random_state=1).split
(x_train,target.values))[0]
```

In [0]:

```
#parameter tuning
#https://scikit-optimize.github.io/#skopt.BayesSearchCV
opt = BayesSearchCV(
    lgb.LGBMClassifier(objective='binary',metric='auc'),
    {
        'num_leaves':Integer(8,15),
        'bagging_fraction':Real(0.3,0.4),
        'feature_fraction':Real(0.3,0.4),
        'learning_rate': (0.01, 0.02),
        'min_data_in_leaf':(80,100),
    },n_jobs=-1, cv=2)

opt.fit(x_train[bay_tr_index],target[bay_tr_index])
print("val. score: %s" % opt.best_score_)
print("test score: %s" % opt.score(x_train[bay_val_index],target[bay_val_index]))
print("Best parameters: ", opt.best_params_)
```

```
val. score: 0.89951
test score: 0.89951
Best parameters:  {'bagging_fraction': 0.3383218916236549, 'feature_fraction': 0.3621429919945276,
'learning_rate': 0.011462880850105828, 'min_data_in_leaf': 98, 'num_leaves': 9}
```

In [0]:

```
#setting the parameters
param = {
    'bagging_freq': 5,
    'bagging_fraction': 0.33,
    'boost_from_average':'false',
    'boost': 'gbdt',
    'feature_fraction': 0.36,
    'learning_rate': 0.011,
```

```
    'max_depth': -1,
    'metric':'auc',
    'min_data_in_leaf':98,
    'min_sum_hessian_in_leaf': 8.0,
    'num_leaves': 9,
    'num_threads':10,
    'tree_learner': 'serial',
    'objective': 'binary',
    'verbosity': 1}
```

In [0]:

```
#https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.train.html#lightgbm.train
#https://www.kaggle.com/ashishpatel26/kfold-lightgbm/code
#(learned from here how to use stratified k-fold with model)

skf = StratifiedKFold(n_splits=10, shuffle=False, random_state=123456)
rem_samp= np.zeros(len(train_df))
pred = np.zeros(len(test_df))

feat_imp_df = pd.DataFrame()

for i,(tr_idx, val_idx) in enumerate(skf.split(x_train,target.values)):
    print("Fold {}".format(i))
    tr_data = lgb.Dataset(x_train[tr_idx], label=target.iloc[tr_idx])
    val_data = lgb.Dataset(x_train[val_idx], label=target.iloc[val_idx])

    num_round = 100000
    clf = lgb.train(param, tr_data, num_round, valid_sets = [tr_data, val_data], verbose_eval=1000,
early_stopping_rounds = 3000)

    rem_samp[val_idx] = clf.predict(x_train[val_idx], num_iteration=clf.best_iteration)

    fold_imp_df = pd.DataFrame()
    fold_imp_df["feature"] = feat
    fold_imp_df["importance"] = clf.feature_importance()
    fold_imp_df["fold"] = i + 1
    feat_imp_df = pd.concat([feat_imp_df,fold_imp_df], axis=0)


    pred+=clf.predict(x_test, num_iteration=clf.best_iteration) /skf.n_splits

print("CV score: {}".format(roc_auc_score(target,rem_samp)))
```

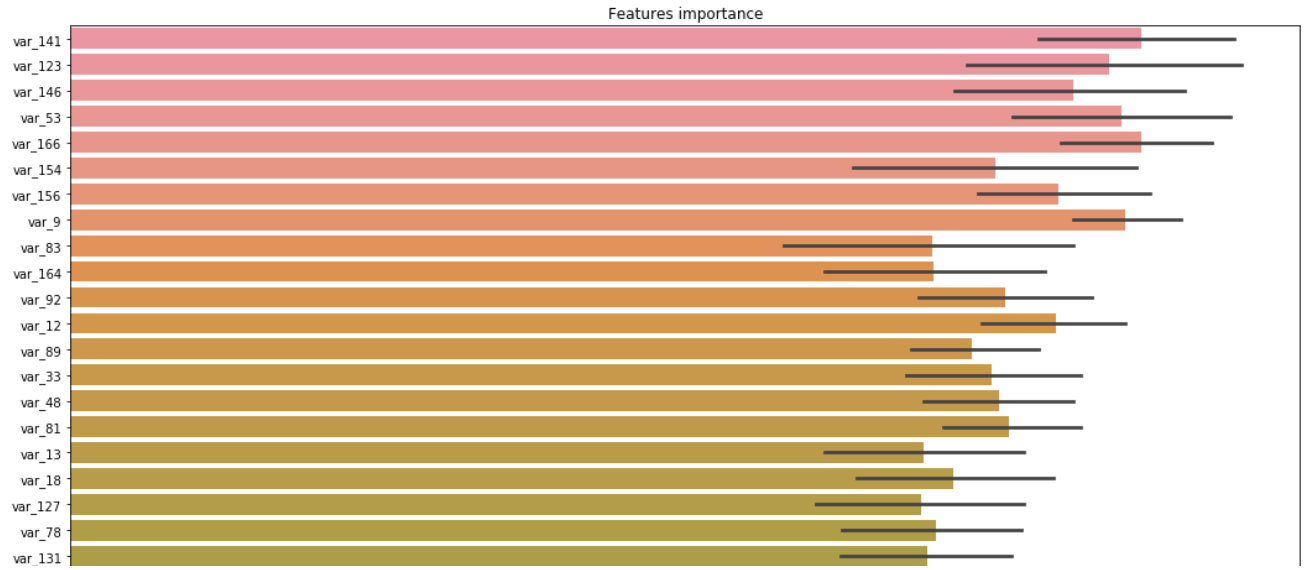
```
Fold 0
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.949189 valid_1's auc: 0.887325
[2000] training's auc: 0.975719 valid_1's auc: 0.880059
[3000] training's auc: 0.990332 valid_1's auc: 0.876435
Early stopping, best iteration is:
[623] training's auc: 0.934268 valid_1's auc: 0.889751
Fold 1
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948619 valid_1's auc: 0.889795
[2000] training's auc: 0.975644 valid_1's auc: 0.882028
[3000] training's auc: 0.990305 valid_1's auc: 0.878322
Early stopping, best iteration is:
[717] training's auc: 0.938311 valid_1's auc: 0.891674
Fold 2
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948994 valid_1's auc: 0.882525
[2000] training's auc: 0.975836 valid_1's auc: 0.872997
[3000] training's auc: 0.989994 valid_1's auc: 0.869132
Early stopping, best iteration is:
[714] training's auc: 0.938448 valid_1's auc: 0.883551
Fold 3
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948795 valid_1's auc: 0.886462
[2000] training's auc: 0.975459 valid_1's auc: 0.878883
[3000] training's auc: 0.990189 valid_1's auc: 0.874536
Early stopping, best iteration is:
[516] training's auc: 0.929154 valid_1's auc: 0.888501
Fold 4
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.949018 valid_1's auc: 0.886522
[2000] training's auc: 0.97508 valid_1's auc: 0.879315
```

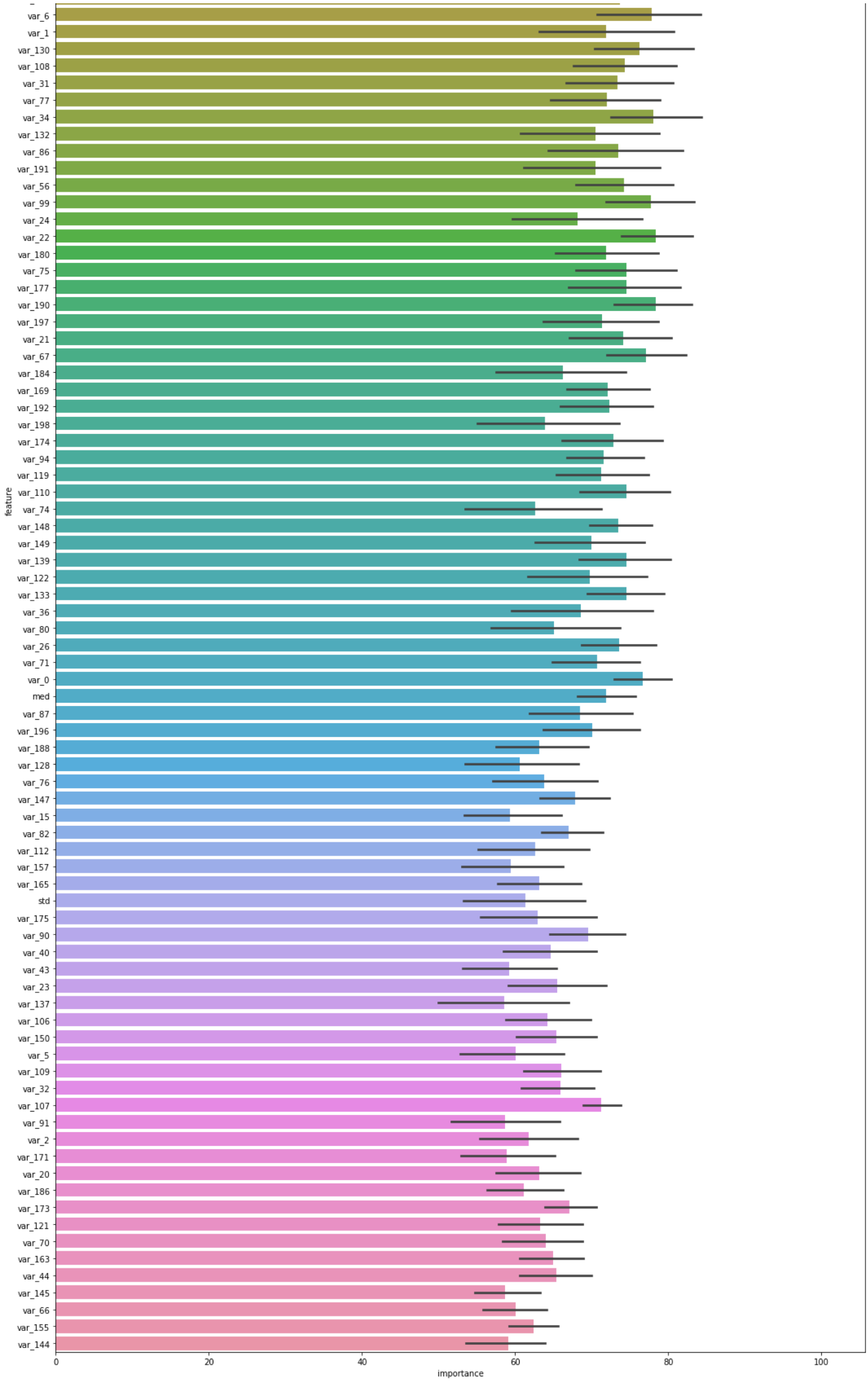
```
[2000] training's auc: 0.97508 valid_1's auc: 0.879315
[3000] training's auc: 0.989587 valid_1's auc: 0.875398
Early stopping, best iteration is:
[539] training's auc: 0.930344 valid_1's auc: 0.889916
Fold 5
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948401 valid_1's auc: 0.89237
[2000] training's auc: 0.975359 valid_1's auc: 0.885908
[3000] training's auc: 0.989917 valid_1's auc: 0.879863
Early stopping, best iteration is:
[734] training's auc: 0.938868 valid_1's auc: 0.894548
Fold 6
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948362 valid_1's auc: 0.887891
[2000] training's auc: 0.974872 valid_1's auc: 0.880389
[3000] training's auc: 0.989775 valid_1's auc: 0.875359
Early stopping, best iteration is:
[546] training's auc: 0.930372 valid_1's auc: 0.891944
Fold 7
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948643 valid_1's auc: 0.889031
[2000] training's auc: 0.975594 valid_1's auc: 0.883167
[3000] training's auc: 0.990296 valid_1's auc: 0.877209
Early stopping, best iteration is:
[698] training's auc: 0.937597 valid_1's auc: 0.89148
Fold 8
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948323 valid_1's auc: 0.891788
[2000] training's auc: 0.975315 valid_1's auc: 0.883349
[3000] training's auc: 0.989951 valid_1's auc: 0.878602
Early stopping, best iteration is:
[516] training's auc: 0.928735 valid_1's auc: 0.894944
Fold 9
Training until validation scores don't improve for 3000 rounds.
[1000] training's auc: 0.948477 valid_1's auc: 0.890356
[2000] training's auc: 0.974977 valid_1's auc: 0.884156
[3000] training's auc: 0.989862 valid_1's auc: 0.876979
Early stopping, best iteration is:
[726] training's auc: 0.938279 valid_1's auc: 0.891647
CV score: 0.8905348564299831
```

In [0]:

```
col= (feat_imp_df[["feature", "importance"]]
        .groupby("feature")
        .mean()
        .sort_values(by="importance", ascending=False)[:100].index)
best_feat = feat_imp_df.loc[feat_imp_df.feature.isin(col)]

plt.figure(figsize=(15,30))
sns.barplot(x="importance", y="feature",
data=best_feat.sort_values(by="importance",ascending=False))
plt.title('Features importance')
plt.tight_layout()
```




feature
var_6
var_1
var_130
var_108
var_31
var_77
var_34
var_132
var_86
var_191
var_56
var_99
var_24
var_22
var_180
var_75
var_177
var_190
var_197
var_21
var_67
var_184
var_169
var_192
var_198
var_174
var_94
var_119
var_110
var_74
var_148
var_149
var_139
var_122
var_133
var_36
var_80
var_26
var_71
var_0
med
var_87
var_196
var_188
var_128
var_76
var_147
var_15
var_82
var_112
var_157
var_165
std
var_175
var_90
var_40
var_43
var_23
var_137
var_106
var_150
var_5
var_109
var_32
var_107
var_91
var_2
var_171
var_20
var_186
var_173
var_121
var_70
var_163
var_44
var_145
var_66
var_155
var_144
0
20
40
60
80
100
importance

In [0]:

```
sub = pd.DataFrame({"ID_code":test_df["ID_code"].values})
sub["target"] = predictions
sub.to_csv("submission.csv", index=False)
```

In [0]:

```
sub=pd.read_csv('submission.csv')
sub.head(10)
```

Out[0]:

	ID_code	target
0	test_0	0.098795
1	test_1	0.212387
2	test_2	0.226738
3	test_3	0.190676
4	test_4	0.037737
5	test_5	0.001349
6	test_6	0.004881
7	test_7	0.174774
8	test_8	0.001927
9	test_9	0.005255

Conclusion

In [0]:

```
from prettytable import PrettyTable
x = PrettyTable()

x.field_names = ["Model", "Train_AUC", "Test_AUC"]

x.add_row(["LGB", 0.89,0.90006])

print(x)
```

```
+-------+-----------+----------+
| Model | Train_AUC | Test_AUC |
+-------+-----------+----------+
|  LGB  |    0.89   | 0.90006  |
+-------+-----------+----------+
```

Steps used-

1-Load the dataset.

2-Perform EDA to understand the data.

3-Do some preprocessing.

4-Try various models.

5-Report AUC.