2_FeatureEngg

April 20, 2020

1 Feature Engineering

1. We will observe the plot of actual feature and their transformed feature engineering
2. Remember! We are not transforming right now. We are just visualizing how data feature transformed after applying featuring.
3. We havr applied feature engineering while we do modelling, so this notebook is for visualizzation purpose only

2 Import Necessary Libraries

[1]:
```
# For column stack
import numpy as np
# To read file
import pandas as pd
# For plotting purpose
import matplotlib.pyplot as plt
import seaborn as sns
```

3 Read train data

[2]:
```
# Locate parent directory
data_dir = "./"

# Read csv file and display top 5 rows
df_train = pd.read_csv(data_dir+'/train.csv')
df_train.head(5)
```

[2]:

	id	target	0	1	2	3	4	5	6	7	…	\
0	0	1.0	-0.098	2.165	0.681	-0.614	1.309	-0.455	-0.236	0.276	…	
1	1	0.0	1.081	-0.973	-0.383	0.326	-0.428	0.317	1.172	0.352	…	
2	2	1.0	-0.523	-0.089	-0.348	0.148	-0.022	0.404	-0.023	-0.172	…	
3	3	1.0	0.067	-0.021	0.392	-1.637	-0.446	-0.725	-1.035	0.834	…	
4	4	1.0	2.347	-0.831	0.511	-0.021	1.225	1.594	0.585	1.509	…	

	290	291	292	293	294	295	296	297	298	299
0	0.867	1.347	0.504	-0.649	0.672	-2.097	1.051	-0.414	1.038	-1.065

```
1 -0.165 -1.695 -1.257  1.359 -0.808 -1.624 -0.458 -1.099 -0.936  0.973
2  0.013  0.263 -1.222  0.726  1.444 -1.165 -1.544  0.004  0.800 -1.211
3 -0.404  0.640 -0.595 -0.966  0.900  0.467 -0.562 -0.254 -0.533  0.238
4  0.898  0.134  2.415 -0.996 -1.006  1.378  1.246  1.478  0.428  0.253

[5 rows x 302 columns]
```

[3]:
```
# Create duplicate df_train
temp = df_train.drop(['id','target'], axis=1)
temp.head()
```

[3]:
```
       0      1      2      3      4      5      6      7      8      9  …  \
0 -0.098  2.165  0.681 -0.614  1.309 -0.455 -0.236  0.276 -2.246  1.825  …
1  1.081 -0.973 -0.383  0.326 -0.428  0.317  1.172  0.352  0.004 -0.291  …
2 -0.523 -0.089 -0.348  0.148 -0.022  0.404 -0.023 -0.172  0.137  0.183  …
3  0.067 -0.021  0.392 -1.637 -0.446 -0.725 -1.035  0.834  0.503  0.274  …
4  2.347 -0.831  0.511 -0.021  1.225  1.594  0.585  1.509 -0.012  2.198  …

     290    291    292    293    294    295    296    297    298    299
0  0.867  1.347  0.504 -0.649  0.672 -2.097  1.051 -0.414  1.038 -1.065
1 -0.165 -1.695 -1.257  1.359 -0.808 -1.624 -0.458 -1.099 -0.936  0.973
2  0.013  0.263 -1.222  0.726  1.444 -1.165 -1.544  0.004  0.800 -1.211
3 -0.404  0.640 -0.595 -0.966  0.900  0.467 -0.562 -0.254 -0.533  0.238
4  0.898  0.134  2.415 -0.996 -1.006  1.378  1.246  1.478  0.428  0.253

[5 rows x 300 columns]
```

4 Feature Engineering

4.1 Mean and Standard deviation value of each row

[4]:
```
df_train['mean'] = np.mean(temp, axis=1)
df_train['std'] = np.std(temp, axis=1)
df_train.head(5)
```

[4]:
```
   id  target      0      1      2      3      4      5      6      7  …  \
0   0     1.0 -0.098  2.165  0.681 -0.614  1.309 -0.455 -0.236  0.276  …
1   1     0.0  1.081 -0.973 -0.383  0.326 -0.428  0.317  1.172  0.352  …
2   2     1.0 -0.523 -0.089 -0.348  0.148 -0.022  0.404 -0.023 -0.172  …
3   3     1.0  0.067 -0.021  0.392 -1.637 -0.446 -0.725 -1.035  0.834  …
4   4     1.0  2.347 -0.831  0.511 -0.021  1.225  1.594  0.585  1.509  …

     292    293    294    295    296    297    298    299      mean       std
0  0.504 -0.649  0.672 -2.097  1.051 -0.414  1.038 -1.065 -0.009223  1.087355
1 -1.257  1.359 -0.808 -1.624 -0.458 -1.099 -0.936  0.973  0.086130  0.984194
2 -1.222  0.726  1.444 -1.165 -1.544  0.004  0.800 -1.211  0.027657  1.011068
3 -0.595 -0.966  0.900  0.467 -0.562 -0.254 -0.533  0.238  0.088357  0.938176
```

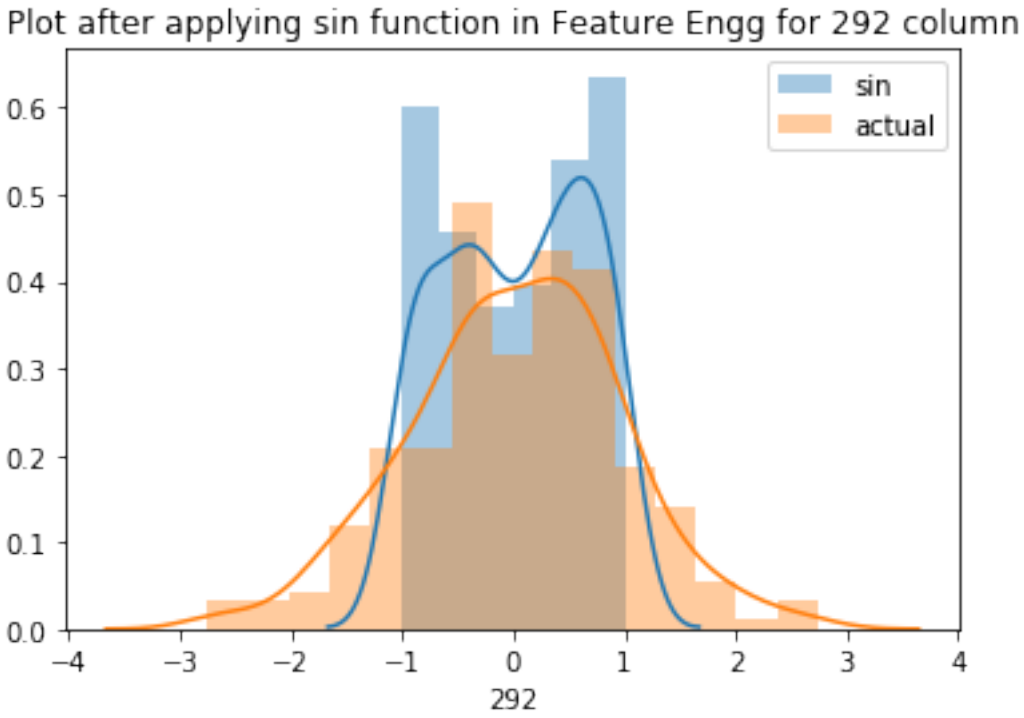
```
4   2.415 -0.996 -1.006   1.378   1.246   1.478   0.428   0.253   0.134413   0.939707

[5 rows x 304 columns]
```

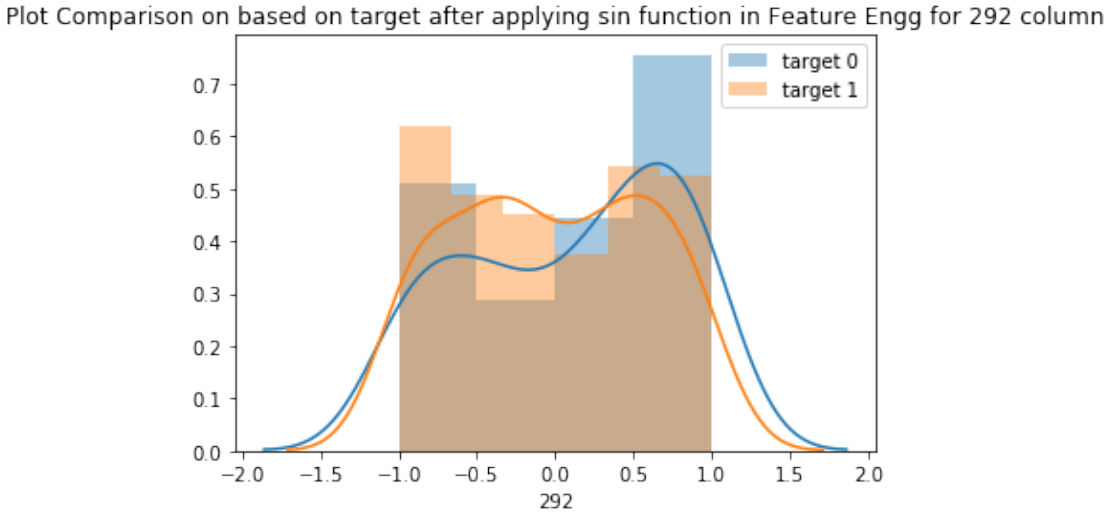
4.2 Trigonmetric function

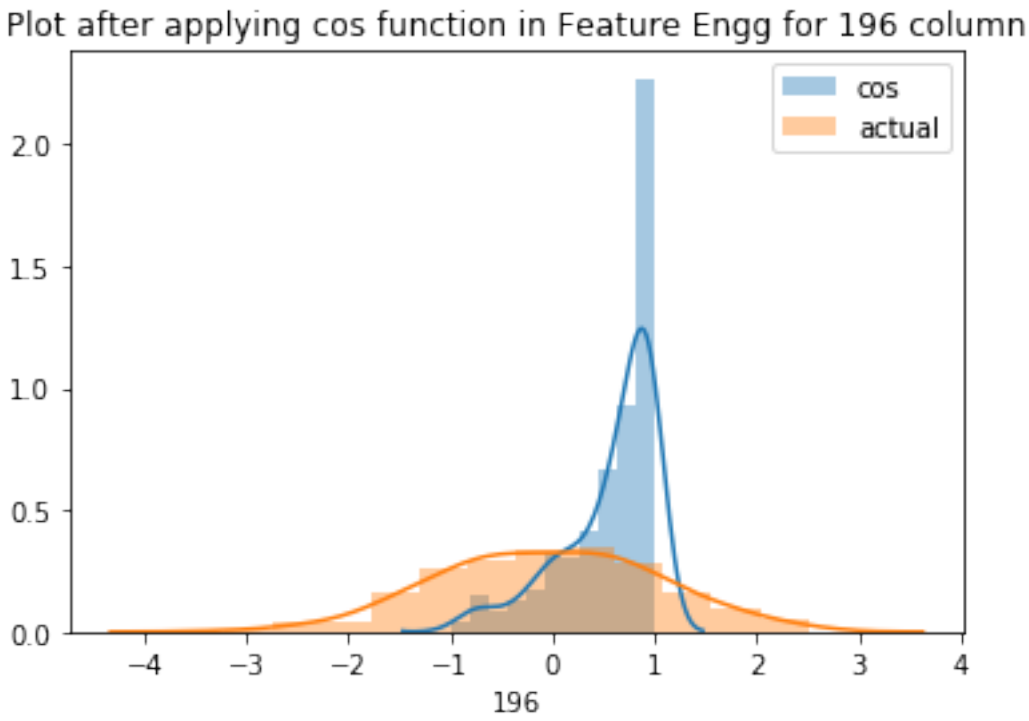
[5]:
```
# Trigonmetrics function (Ref Docs: https://docs.scipy.org/doc/numpy/reference/
 ↪routines.math.html)

sin_temp = np.sin(temp)
cos_temp = np.cos(temp)
tan_temp = np.tan(temp)
```

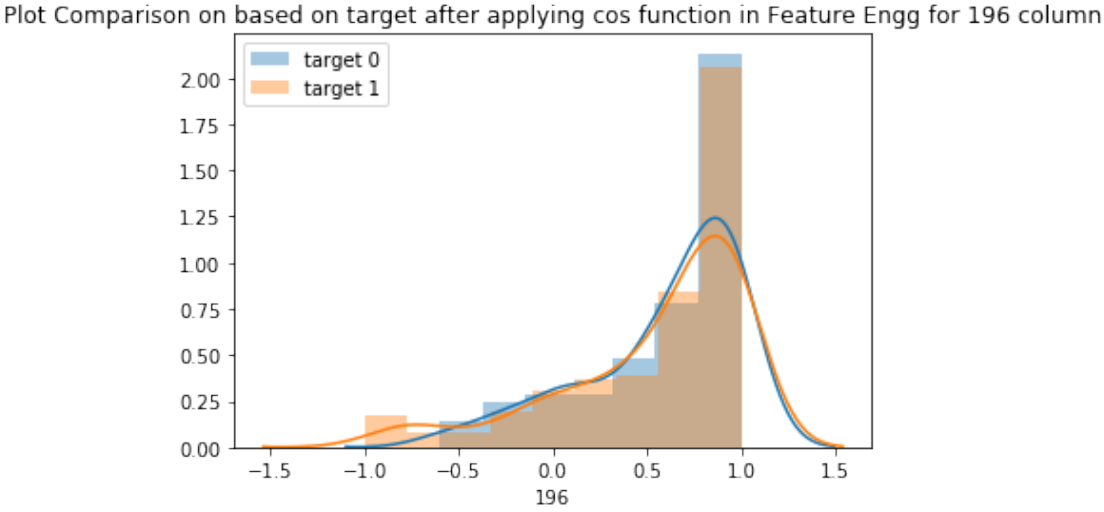
[6]:
```
# Hows its look like in visual way

# Create a function to plot the graph of actual plot vs after feature engg␣
 ↪applied
def visual_fe(fe_name, fe_var):
    '''
    Parameter:
    fe_name: name of transformation Feature engg (string)
    fe_var: data after applying feature engineering

    Return:
    Plot 2 graphs.
    First plot for transformation of actual plot and after applying feature␣
 ↪engg plot for any 'r' column
    Second plot for showing the plot of applying feature engg on the based on␣
 ↪target value for any 'r' column
    '''
    r = str(np.random.randint(0,300))

    plt.figure(1)
    sns.distplot(fe_var[r], label=fe_name)
    sns.distplot(df_train[r], label='actual')
    plt.title('Plot after applying {} function in Feature Engg for {} column'.
 ↪format(fe_name,r))
    plt.legend()

    plt.figure(2)
    sns.distplot(fe_var[df_train['target']==0][r], label='target 0')
    sns.distplot(fe_var[df_train['target']==1][r], label='target 1')
    plt.title('Plot Comparison on based on target after applying {} function in␣
 ↪Feature Engg for {} column'.format(fe_name,r))
    plt.legend()
```

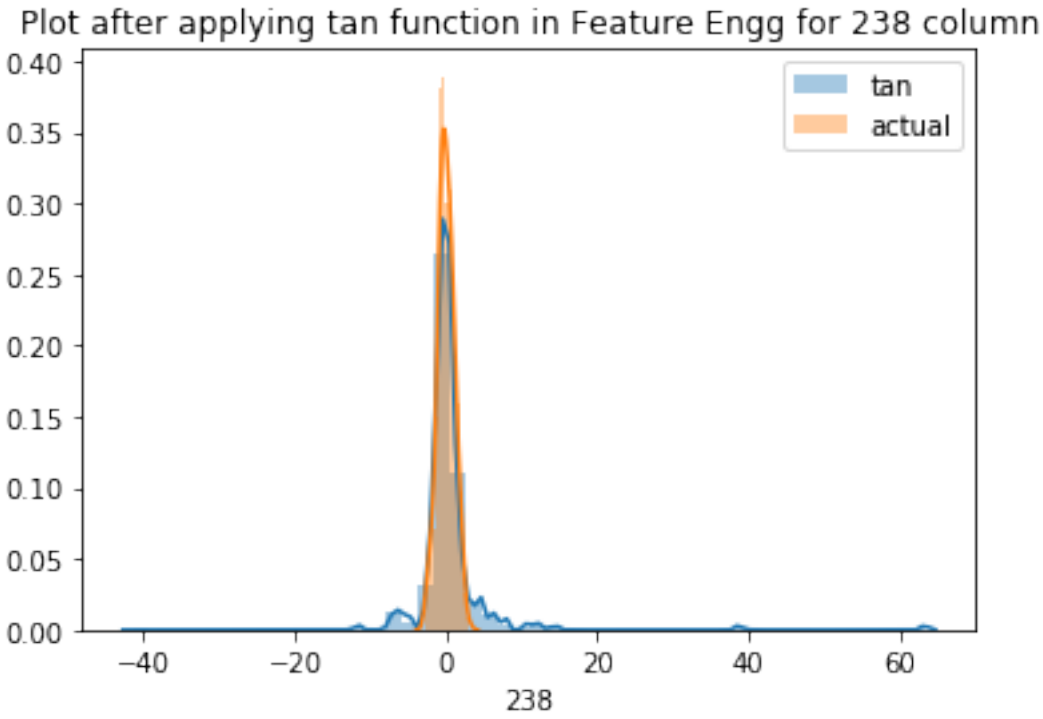
```
    plt.show()

visual_fe('sin',sin_temp)
visual_fe('cos',cos_temp)
visual_fe('tan',tan_temp)
```

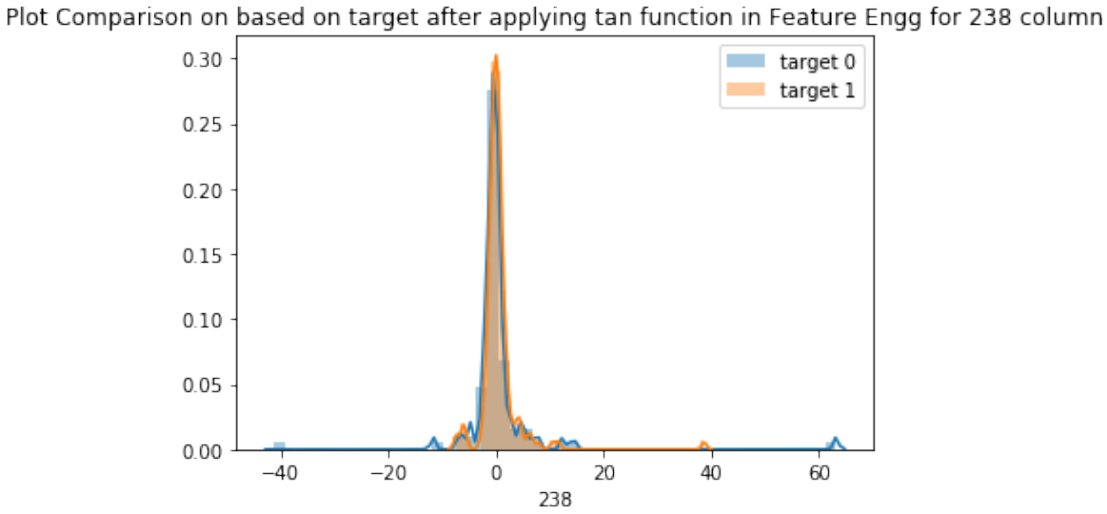





Plot after applying cos function in Feature Engg for 196 column
cos
actual
2.0
1.5
1.0
0.5
0.0
−4
−3
−2
−1
0
1
2
3
4
196


Plot Comparison on based on target after applying cos function in Feature Engg for 196 column
target 0
target 1
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
196





Compute all sine value of all given value [0-299] columns then we take means of each column. Similarly same for other trigometric function

[7]:
```
df_train['mean_sin'] = np.mean(sin_temp, axis=1)
df_train['mean_cos'] = np.mean(cos_temp, axis=1)
df_train['mean_tan'] = np.mean(tan_temp, axis=1)
```

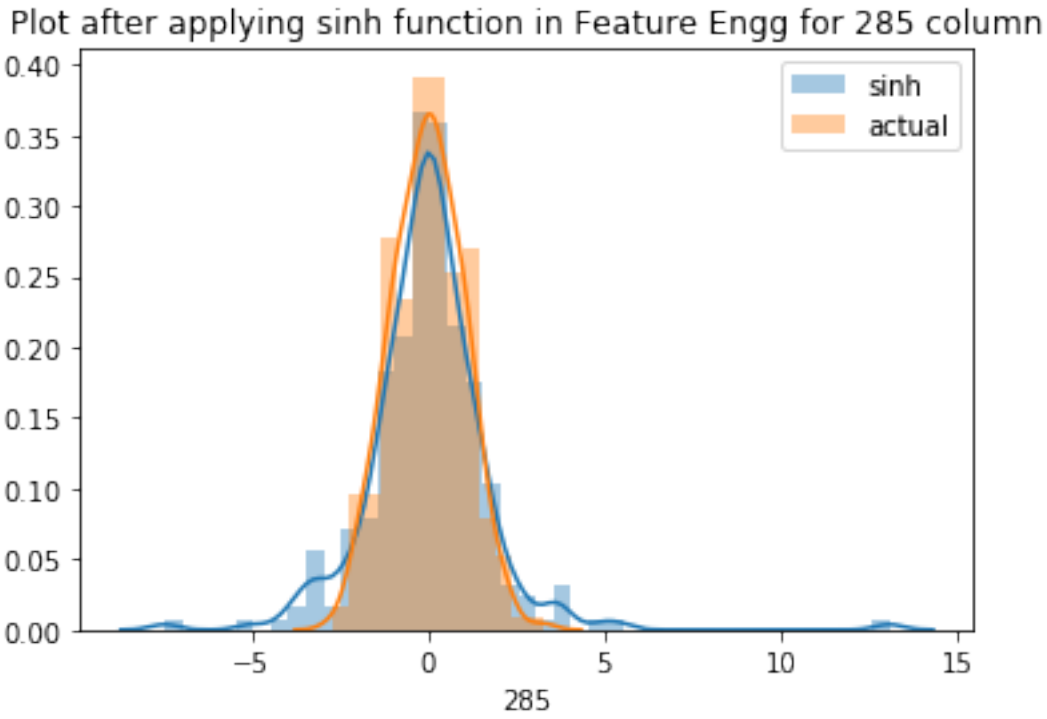
```
df_train.head(5)
```

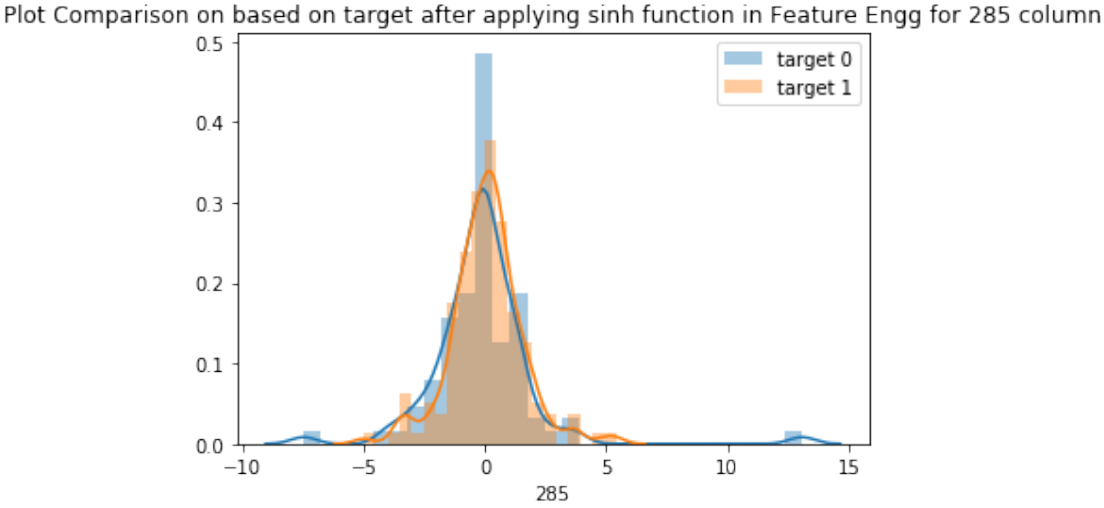
[7]:
```
   id  target      0      1      2      3      4      5      6      7  …  \
0   0     1.0 -0.098  2.165  0.681 -0.614  1.309 -0.455 -0.236  0.276  …
1   1     0.0  1.081 -0.973 -0.383  0.326 -0.428  0.317  1.172  0.352  …
2   2     1.0 -0.523 -0.089 -0.348  0.148 -0.022  0.404 -0.023 -0.172  …
3   3     1.0  0.067 -0.021  0.392 -1.637 -0.446 -0.725 -1.035  0.834  …
4   4     1.0  2.347 -0.831  0.511 -0.021  1.225  1.594  0.585  1.509  …

     295    296    297    298    299      mean       std  mean_sin  mean_cos  \
0 -2.097  1.051 -0.414  1.038 -1.065 -0.009223  1.087355 -0.010536  0.537968
1 -1.624 -0.458 -1.099 -0.936  0.973  0.086130  0.984194  0.075490  0.611600
2 -1.165 -1.544  0.004  0.800 -1.211  0.027657  1.011068 -0.005509  0.599358
3  0.467 -0.562 -0.254 -0.533  0.238  0.088357  0.938176  0.046067  0.645721
4  1.378  1.246  1.478  0.428  0.253  0.134413  0.939707  0.059548  0.643508

   mean_tan
0 -0.315591
1  0.607457
2  0.104777
3  0.891722
4  0.274261

[5 rows x 307 columns]
```

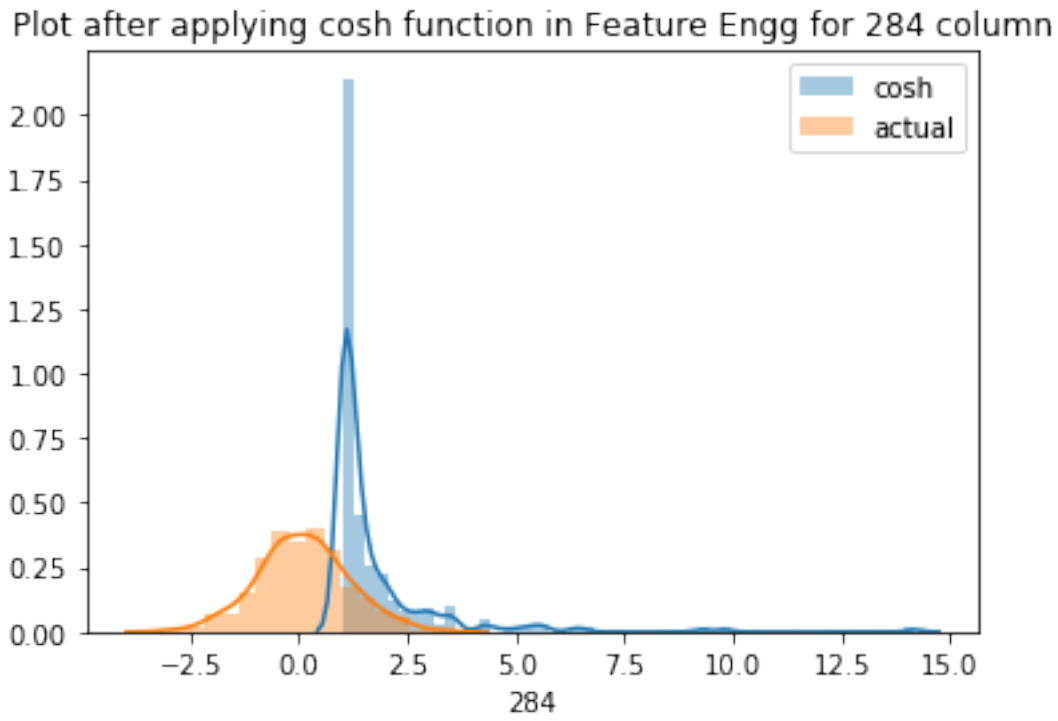
4.3 Hyberbolic Function

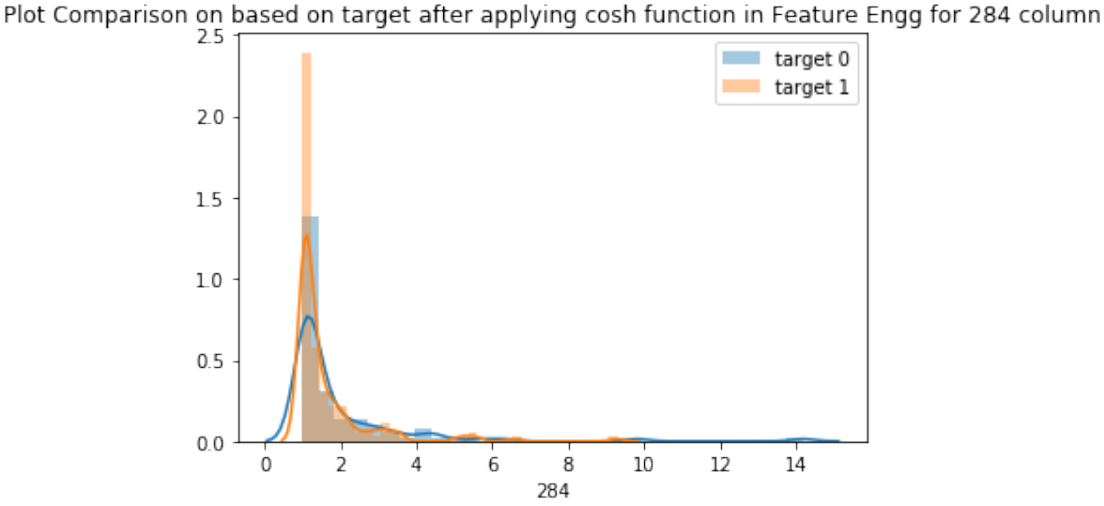
[8]:
```
# Hyperbolic function (Ref Docs: https://docs.scipy.org/doc/numpy/reference/
 ↪routines.math.html)

sinh_temp = np.sinh(temp)
cosh_temp = np.cosh(temp)
tanh_temp = np.tanh(temp)
```

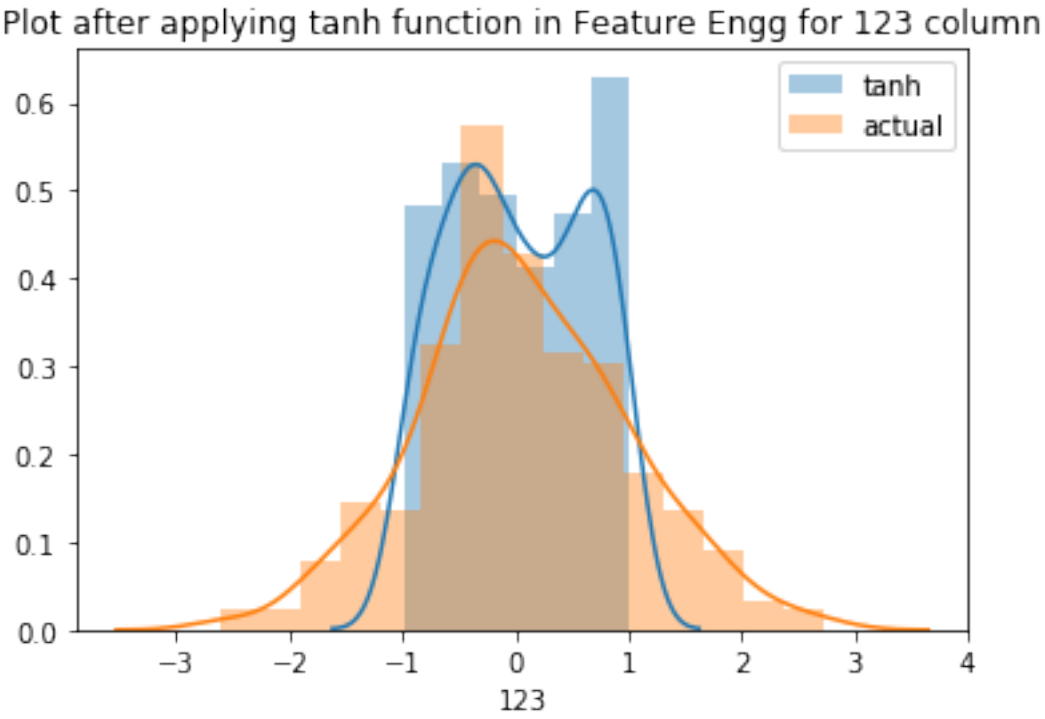
[9]:
```
visual_fe('sinh',sinh_temp)
visual_fe('cosh',cosh_temp)
visual_fe('tanh',tanh_temp)
```

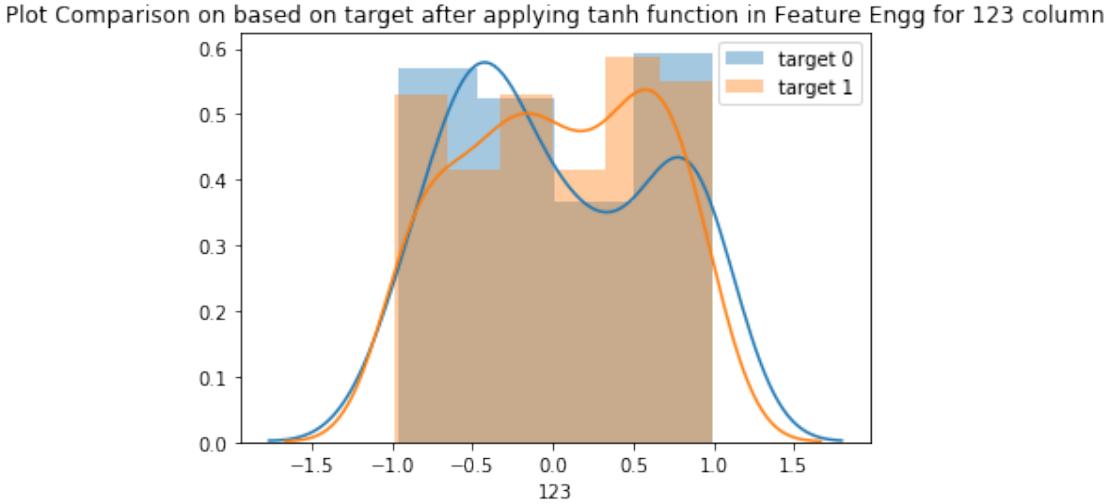

Plot after applying sinh function in Feature Engg for 285 column
sinh
actual
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−5
0
5
10
15
285


Plot Comparison on based on target after applying sinh function in Feature Engg for 285 column
target 0
target 1
0.5
0.4
0.3
0.2
0.1
0.0
−10
−5
0
5
10
15
285


Plot after applying cosh function in Feature Engg for 284 column
cosh
actual
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
284


Plot Comparison on based on target after applying cosh function in Feature Engg for 284 column
target 0
target 1
2.5
2.0
1.5
1.0
0.5
0.0
0
2
4
6
8
10
12
14
284





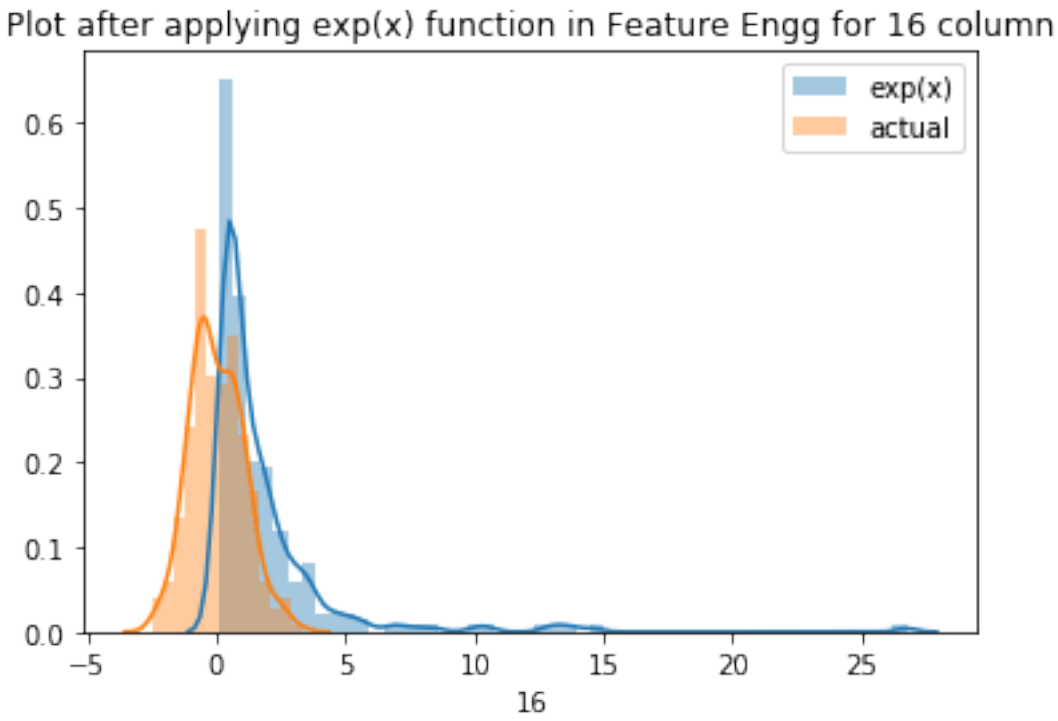
[10]:
```
df_train['mean_sinh'] = np.mean(sin_temp, axis=1)
df_train['mean_cosh'] = np.mean(cos_temp, axis=1)
df_train['mean_tanh'] = np.mean(tan_temp, axis=1)
df_train.head(5)
```

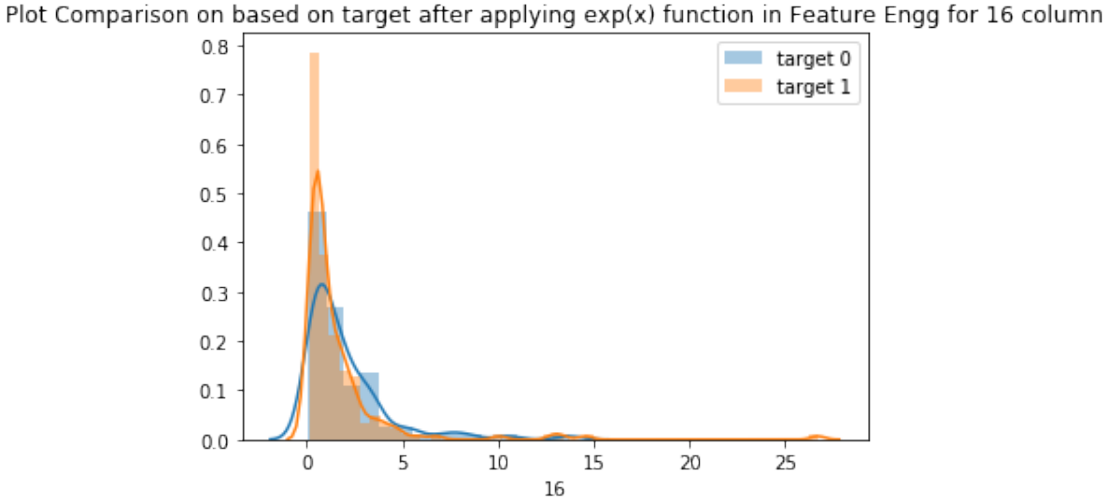
[10]:
```
   id  target      0      1      2      3      4      5      6      7  …  \
0   0     1.0 -0.098  2.165  0.681 -0.614  1.309 -0.455 -0.236  0.276  …
1   1     0.0  1.081 -0.973 -0.383  0.326 -0.428  0.317  1.172  0.352  …
2   2     1.0 -0.523 -0.089 -0.348  0.148 -0.022  0.404 -0.023 -0.172  …
3   3     1.0  0.067 -0.021  0.392 -1.637 -0.446 -0.725 -1.035  0.834  …
4   4     1.0  2.347 -0.831  0.511 -0.021  1.225  1.594  0.585  1.509  …

     298    299      mean       std  mean_sin  mean_cos  mean_tan  mean_sinh  \
0  1.038 -1.065 -0.009223  1.087355 -0.010536  0.537968 -0.315591  -0.010536
1 -0.936  0.973  0.086130  0.984194  0.075490  0.611600  0.607457   0.075490
2  0.800 -1.211  0.027657  1.011068 -0.005509  0.599358  0.104777  -0.005509
3 -0.533  0.238  0.088357  0.938176  0.046067  0.645721  0.891722   0.046067
4  0.428  0.253  0.134413  0.939707  0.059548  0.643508  0.274261   0.059548

   mean_cosh  mean_tanh
0   0.537968  -0.315591
1   0.611600   0.607457
2   0.599358   0.104777
3   0.645721   0.891722
4   0.643508   0.274261

[5 rows x 310 columns]
```

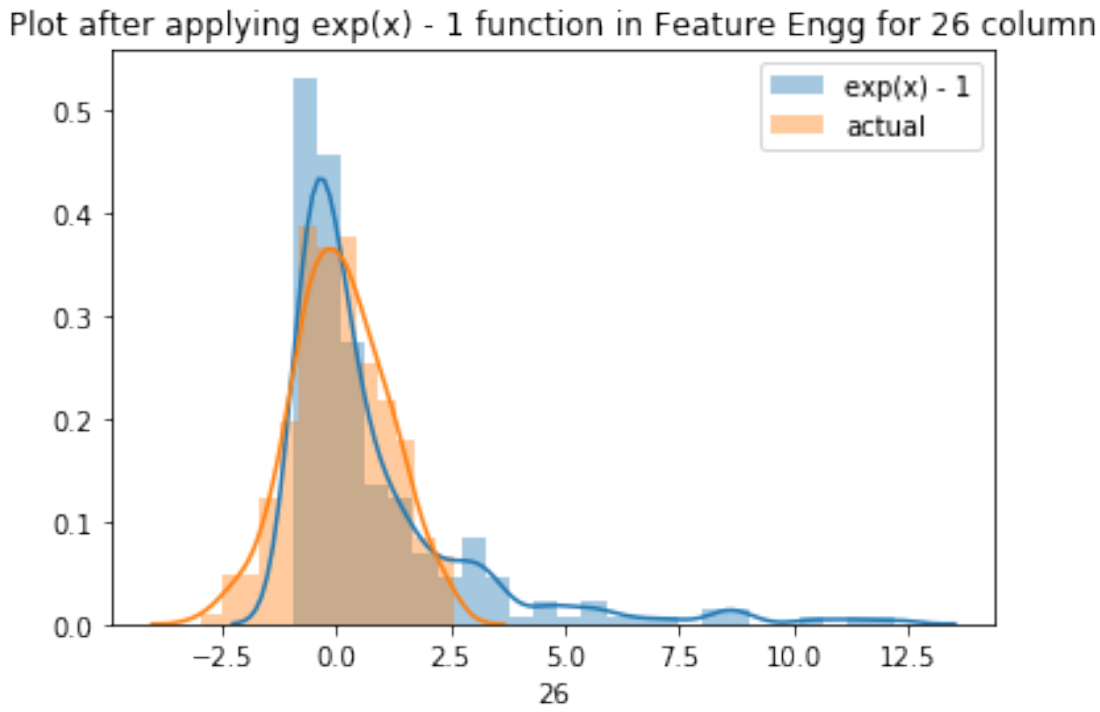
4.4 Exponents

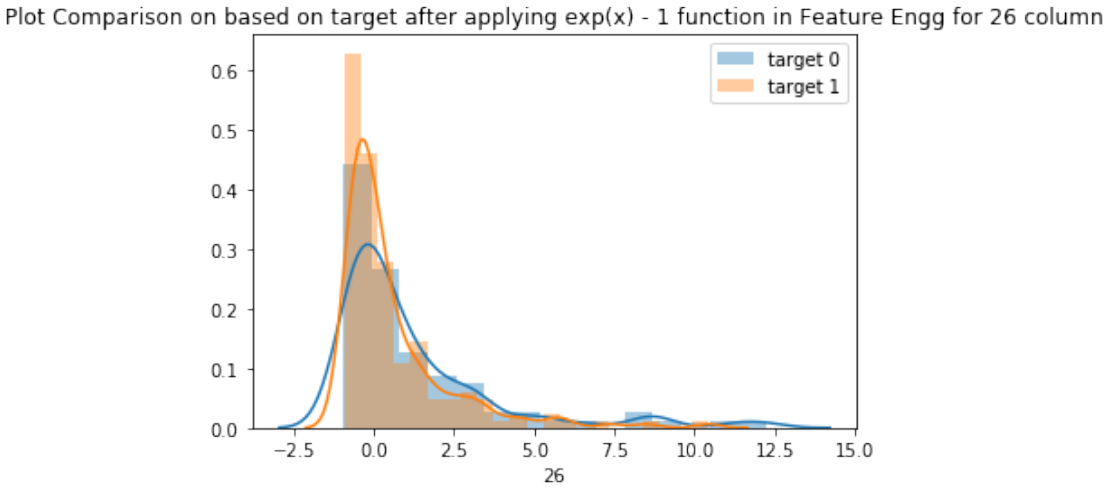
[11]:
```
# Explonents and Logarithm (Ref Docs: https://docs.scipy.org/doc/numpy/
 ↪reference/routines.math.html)

exp_temp = np.exp(temp)
expm1_temp = np.expm1(temp)
exp2_temp = np.exp2(temp)
```

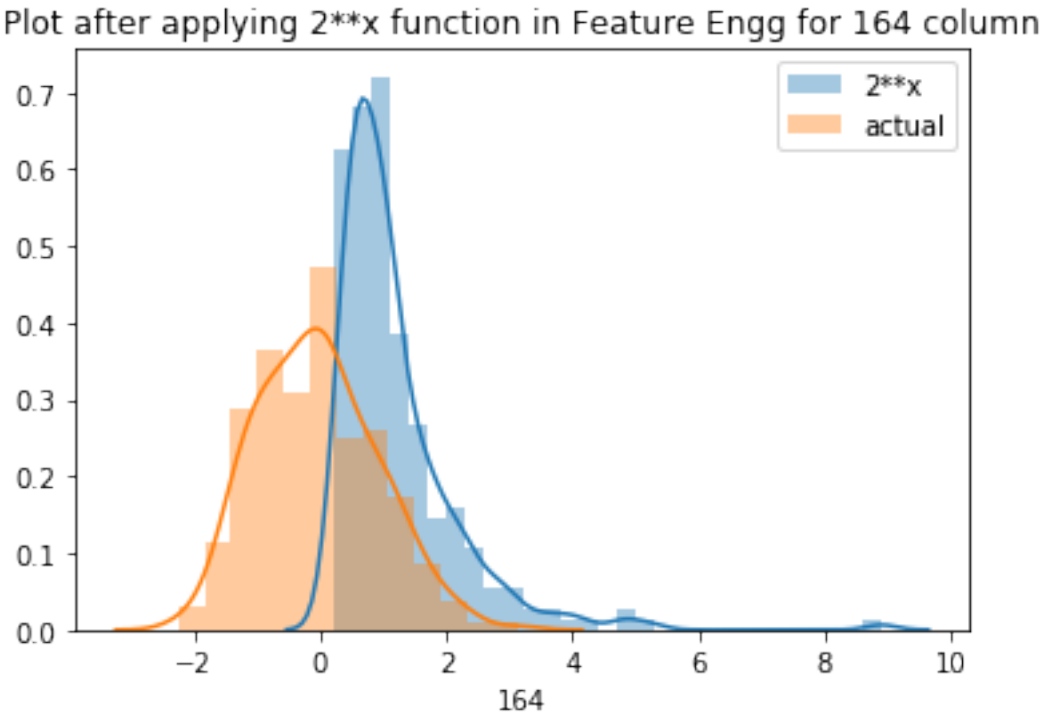
[12]:
```
visual_fe('exp(x)',exp_temp)
visual_fe('exp(x) - 1',expm1_temp)
visual_fe('2**x',exp2_temp)
```

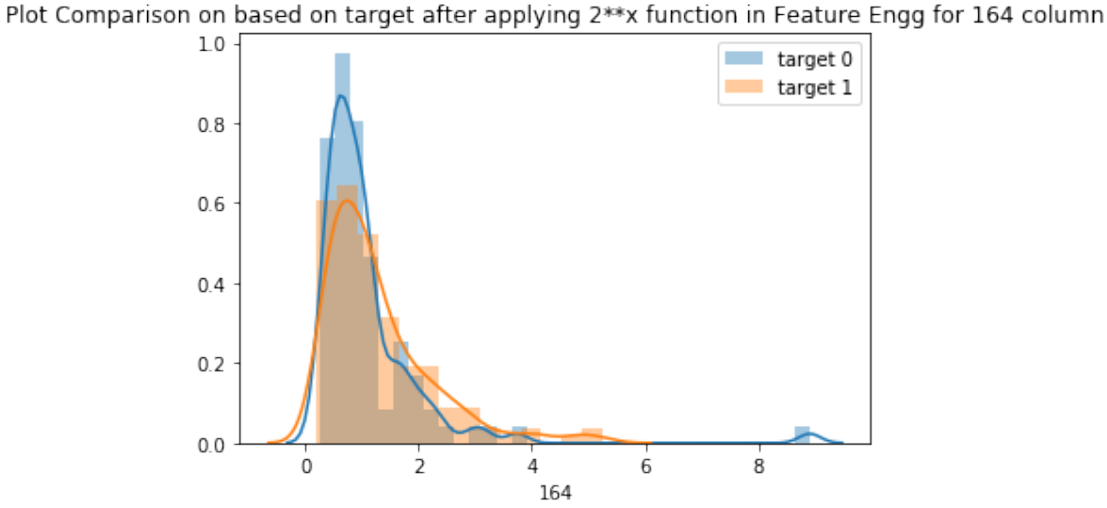

Plot after applying exp(x) function in Feature Engg for 16 column
exp(x)
actual
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−5
0
5
10
15
20
25
16


Plot Comparison on based on target after applying exp(x) function in Feature Engg for 16 column
target 0
target 1
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
25
16


Plot after applying exp(x) - 1 function in Feature Engg for 26 column
exp(x) - 1
actual
0.5
0.4
0.3
0.2
0.1
0.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
26


Plot Comparison on based on target after applying exp(x) - 1 function in Feature Engg for 26 column
target 0
target 1
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
26





[13]:
```
df_train['mean_exp'] = np.mean(exp_temp, axis=1)
df_train['mean_expm1'] = np.mean(expm1_temp, axis=1)
df_train['mean_exp2'] = np.mean(exp2_temp, axis=1)
df_train.head(5)
```

[13]:
```
   id  target      0      1      2      3      4      5      6      7  …  \
0   0     1.0 -0.098  2.165  0.681 -0.614  1.309 -0.455 -0.236  0.276  …
1   1     0.0  1.081 -0.973 -0.383  0.326 -0.428  0.317  1.172  0.352  …
2   2     1.0 -0.523 -0.089 -0.348  0.148 -0.022  0.404 -0.023 -0.172  …
3   3     1.0  0.067 -0.021  0.392 -1.637 -0.446 -0.725 -1.035  0.834  …
4   4     1.0  2.347 -0.831  0.511 -0.021  1.225  1.594  0.585  1.509  …

        std  mean_sin  mean_cos  mean_tan  mean_sinh  mean_cosh  mean_tanh  \
0  1.087355 -0.010536  0.537968 -0.315591  -0.010536   0.537968  -0.315591
1  0.984194  0.075490  0.611600  0.607457   0.075490   0.611600   0.607457
2  1.011068 -0.005509  0.599358  0.104777  -0.005509   0.599358   0.104777
3  0.938176  0.046067  0.645721  0.891722   0.046067   0.645721   0.891722
4  0.939707  0.059548  0.643508  0.274261   0.059548   0.643508   0.274261

   mean_exp  mean_expm1  mean_exp2
0  1.760647    0.760647   1.315869
1  1.712292    0.712292   1.324817
2  1.749107    0.749107   1.313960
3  1.752101    0.752101   1.326229
4  1.861741    0.861741   1.377569

[5 rows x 313 columns]
```

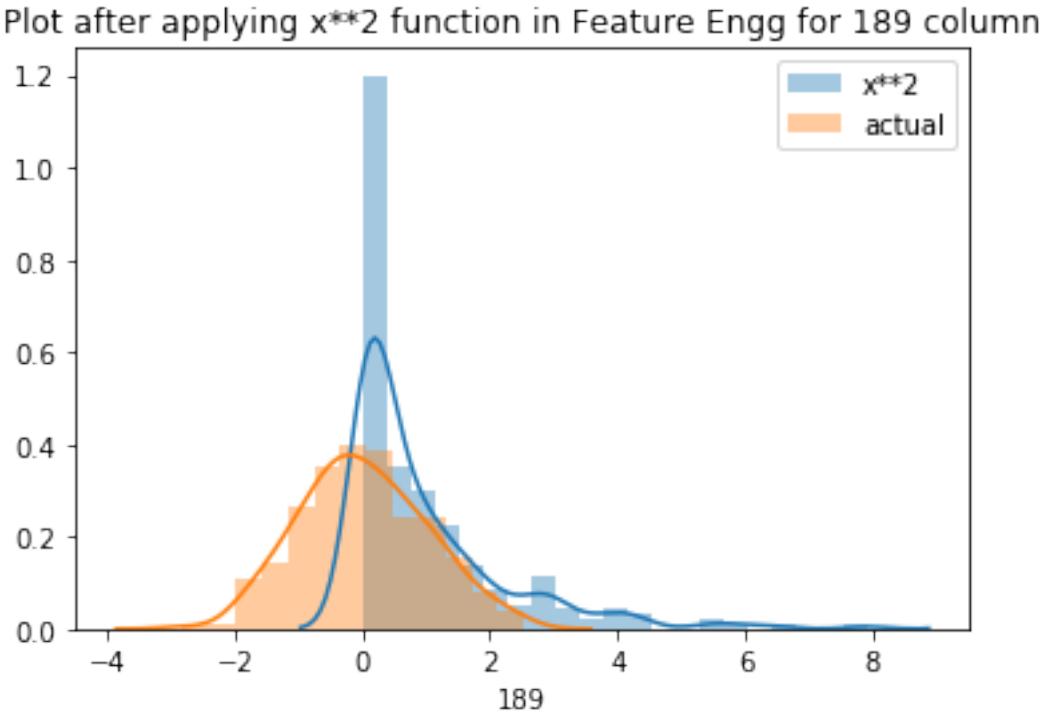
4.5 Some polynomial operation

[14]:
```
# X**2
df_train['mean_x2'] = np.mean(np.power(temp,2), axis=1)

# X**3
df_train['mean_x3'] = np.mean(np.power(temp,3), axis=1)

# X**4
df_train['mean_x4'] = np.mean(np.power(temp,4), axis=1)
df_train.head(5)
```

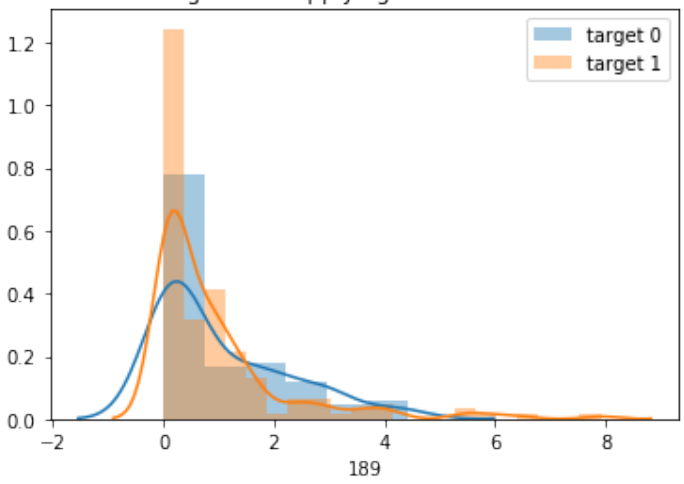
[14]:
```
   id  target      0      1      2      3      4      5      6      7  …  \
0   0     1.0 -0.098  2.165  0.681 -0.614  1.309 -0.455 -0.236  0.276  …
1   1     0.0  1.081 -0.973 -0.383  0.326 -0.428  0.317  1.172  0.352  …
2   2     1.0 -0.523 -0.089 -0.348  0.148 -0.022  0.404 -0.023 -0.172  …
3   3     1.0  0.067 -0.021  0.392 -1.637 -0.446 -0.725 -1.035  0.834  …
4   4     1.0  2.347 -0.831  0.511 -0.021  1.225  1.594  0.585  1.509  …

   mean_tan  mean_sinh  mean_cosh  mean_tanh  mean_exp  mean_expm1  mean_exp2  \
0 -0.315591  -0.010536   0.537968  -0.315591  1.760647    0.760647   1.315869
1  0.607457   0.075490   0.611600   0.607457  1.712292    0.712292   1.324817
2  0.104777  -0.005509   0.599358   0.104777  1.749107    0.749107   1.313960
3  0.891722   0.046067   0.645721   0.891722  1.752101    0.752101   1.326229
```

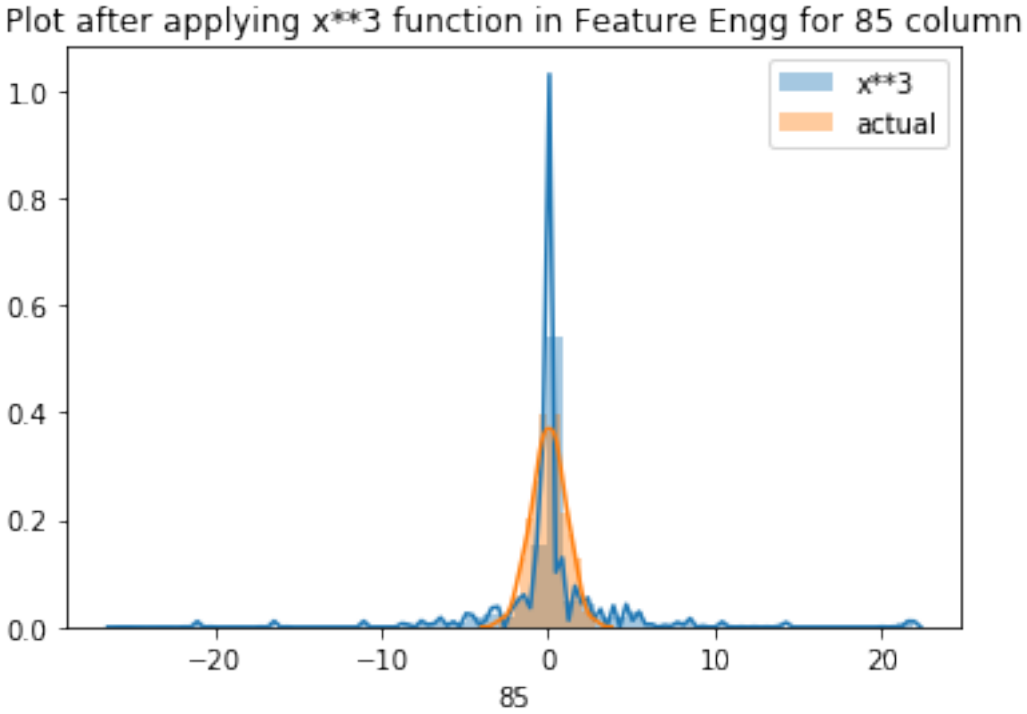
```
4  0.274261   0.059548   0.643508   0.274261  1.861741    0.861741   1.377569

    mean_x2   mean_x3   mean_x4
0  1.182425  0.015243  3.584848
1  0.976056  0.047272  2.766570
2  1.023024  0.266454  3.092631
3  0.887980  0.371308  2.553467
4  0.901115  0.613952  2.671541

[5 rows x 316 columns]
```

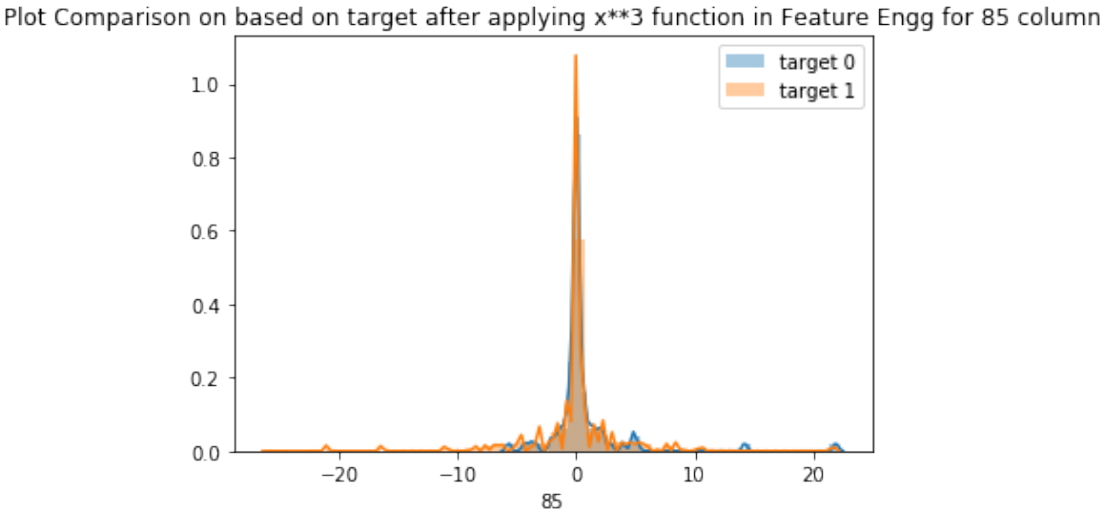
[15]:
```
visual_fe('x**2',np.power(temp,2))
visual_fe('x**3',np.power(temp,3))
visual_fe('x**4',np.power(temp,4))
```

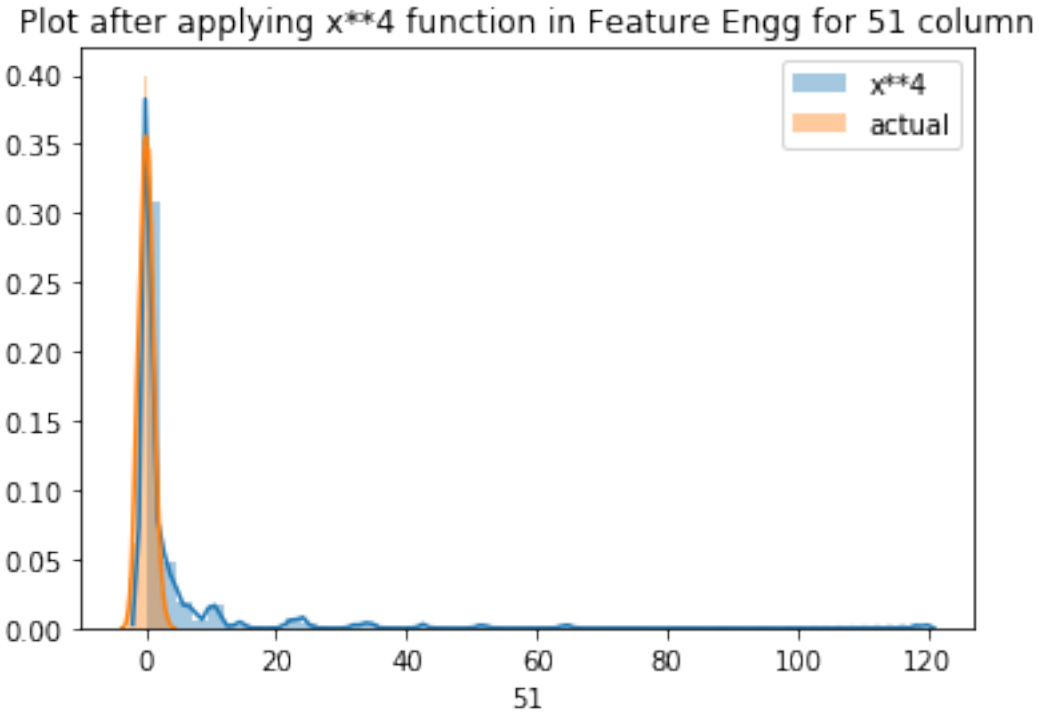


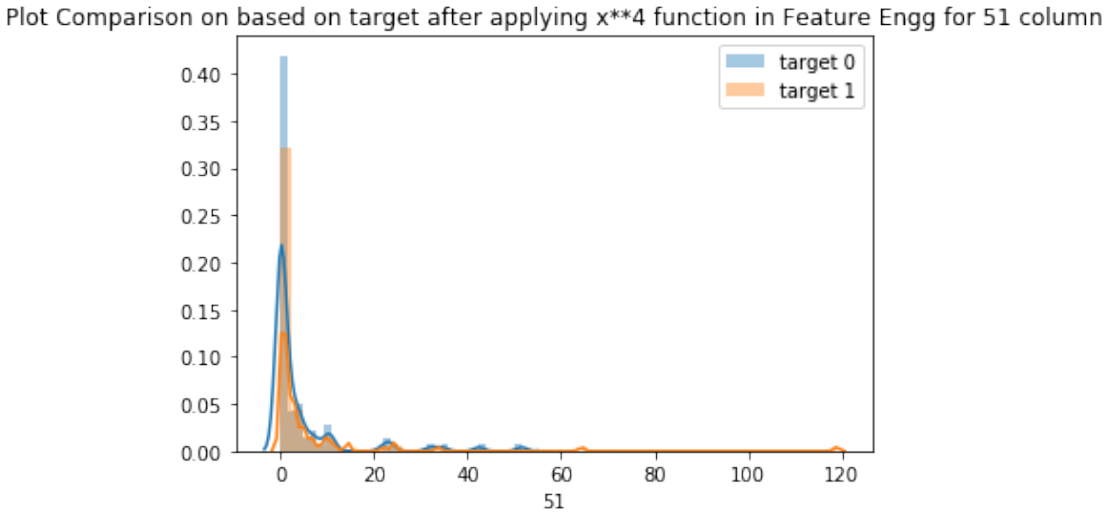
Plot Comparison on based on target after applying x**2 function in Feature Engg for 189 column






Plot Comparison on based on target after applying x**3 function in Feature Engg for 85 column
target 0
target 1
1.0
0.8
0.6
0.4
0.2
0.0
−20
−10
0
10
20
85


Plot after applying x**4 function in Feature Engg for 51 column
x**4
actual
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
120
51



5 Create FE function to combine all into one wrap

[16]:
```
def feature_enng(df):
    temp = df.drop(['id','target'], axis=1)

    # Mean and Std FE
    df['mean'] = np.mean(temp, axis=1)
    df['std'] = np.std(temp, axis=1)

    # Trigometric FE
    sin_temp = np.sin(temp)
    cos_temp = np.cos(temp)
    tan_temp = np.tan(temp)
    df['mean_sin'] = np.mean(sin_temp, axis=1)
    df['mean_cos'] = np.mean(cos_temp, axis=1)
    df['mean_tan'] = np.mean(tan_temp, axis=1)

    # Hyperbolic FE
    sinh_temp = np.sinh(temp)
    cosh_temp = np.cosh(temp)
    tanh_temp = np.tanh(temp)
    df['mean_sinh'] = np.mean(sin_temp, axis=1)
    df['mean_cosh'] = np.mean(cos_temp, axis=1)
    df['mean_tanh'] = np.mean(tan_temp, axis=1)

    # Exponents FE
    exp_temp = np.exp(temp)
    expm1_temp = np.expm1(temp)
```

```
    exp2_temp = np.exp2(temp)
    df['mean_exp'] = np.mean(exp_temp, axis=1)
    df['mean_expm1'] = np.mean(expm1_temp, axis=1)
    df['mean_exp2'] = np.mean(exp2_temp, axis=1)

    # Polynomial FE
    # X**2
    df['mean_x2'] = np.mean(np.power(temp,2), axis=1)
    # X**3
    df['mean_x3'] = np.mean(np.power(temp,3), axis=1)
    # X**4
    df['mean_x4'] = np.mean(np.power(temp,4), axis=1)

    return df
```

[17]:
```
# Read csv file and display top 5 rows
df_train = pd.read_csv(data_dir+'/train.csv')
df_train.head(5)

df_train = feature_enng(df_train)
df_train.head(5)
```

[17]:

	id	target	0	1	2	3	4	5	6	7	…	\
0	0	1.0	-0.098	2.165	0.681	-0.614	1.309	-0.455	-0.236	0.276	…	
1	1	0.0	1.081	-0.973	-0.383	0.326	-0.428	0.317	1.172	0.352	…	
2	2	1.0	-0.523	-0.089	-0.348	0.148	-0.022	0.404	-0.023	-0.172	…	
3	3	1.0	0.067	-0.021	0.392	-1.637	-0.446	-0.725	-1.035	0.834	…	
4	4	1.0	2.347	-0.831	0.511	-0.021	1.225	1.594	0.585	1.509	…	

	mean_tan	mean_sinh	mean_cosh	mean_tanh	mean_exp	mean_expm1	mean_exp2	\
0	-0.315591	-0.010536	0.537968	-0.315591	1.760647	0.760647	1.315869	
1	0.607457	0.075490	0.611600	0.607457	1.712292	0.712292	1.324817	
2	0.104777	-0.005509	0.599358	0.104777	1.749107	0.749107	1.313960	
3	0.891722	0.046067	0.645721	0.891722	1.752101	0.752101	1.326229	
4	0.274261	0.059548	0.643508	0.274261	1.861741	0.861741	1.377569	

	mean_x2	mean_x3	mean_x4
0	1.182425	0.015243	3.584848
1	0.976056	0.047272	2.766570
2	1.023024	0.266454	3.092631
3	0.887980	0.371308	2.553467
4	0.901115	0.613952	2.671541

[5 rows x 316 columns]

6 Summary

1. So, observing after applying feature engineering, we can differentiate target with this feature engineering.
2. Let try with this feature engineering and without featuring engineer impactness of models.

[ ]: