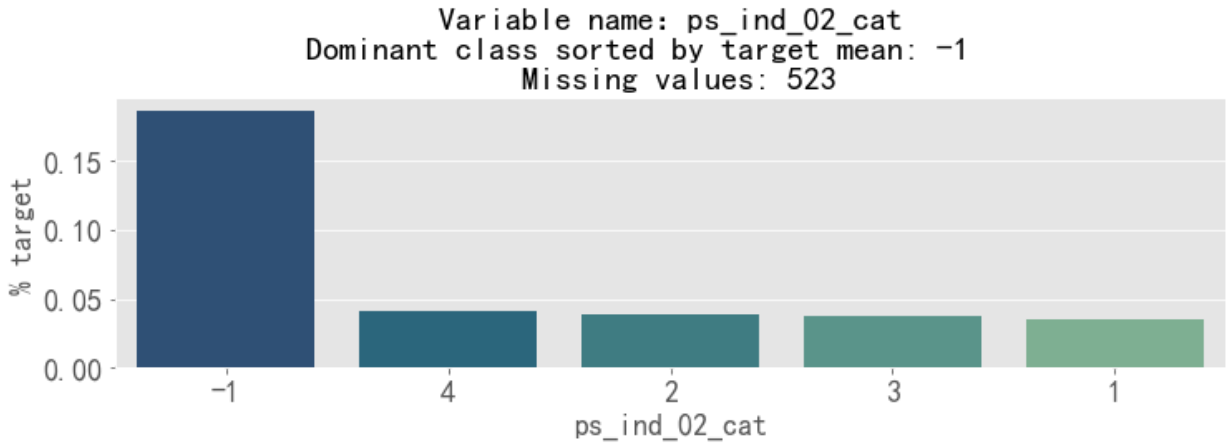
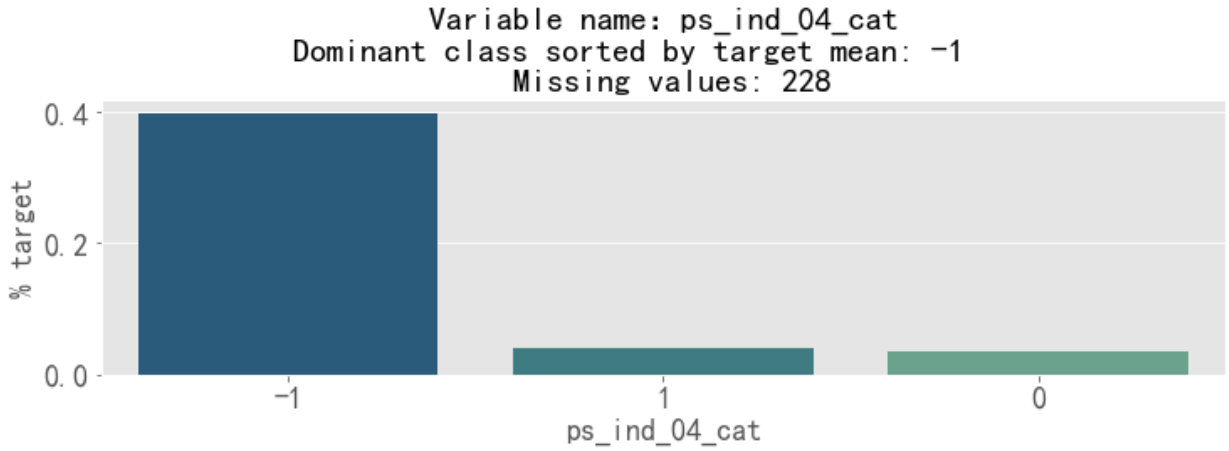
In [5]:
```
from data_management import meta
```

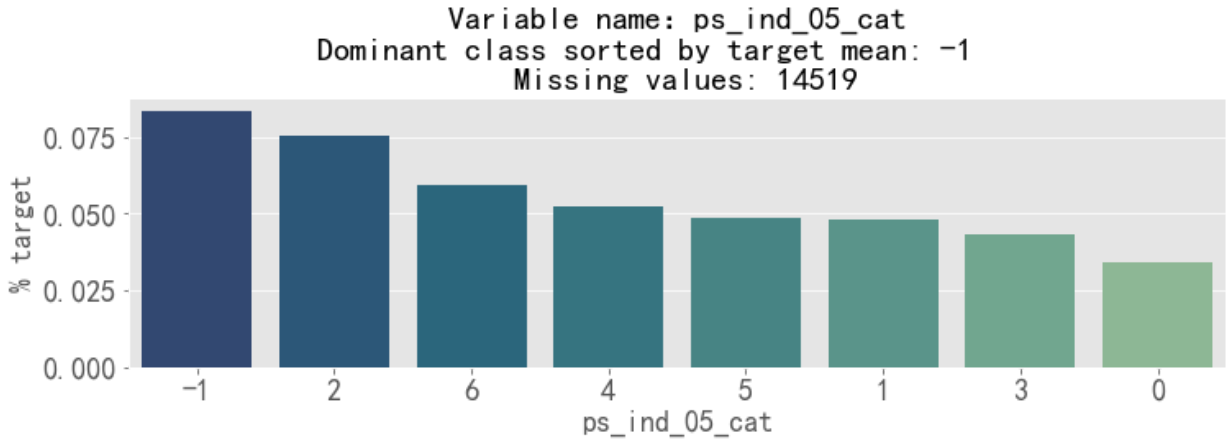
In [6]:
```
metadata = meta(train,test)
```

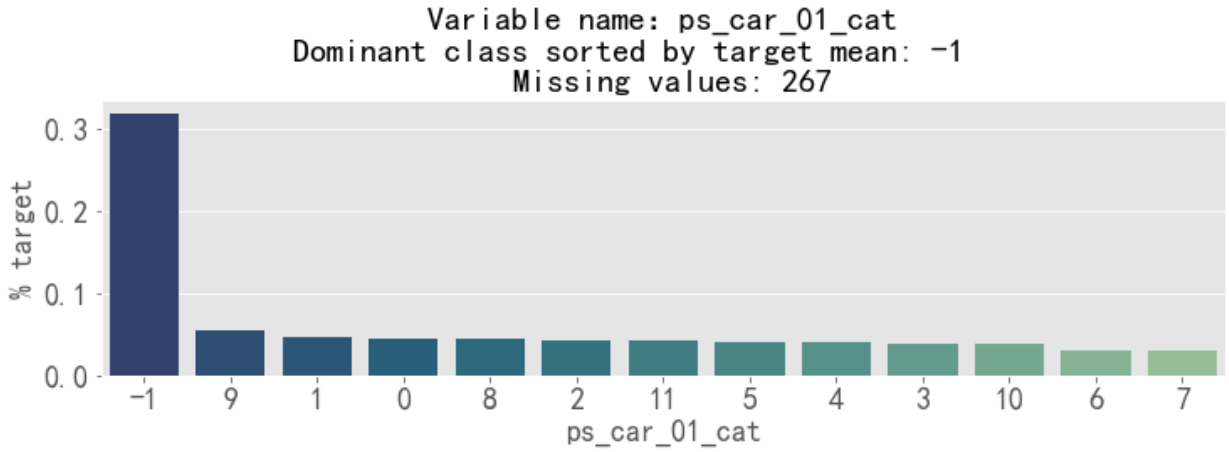
In [8]:
```
cat_cols = metadata[(metadata.level == 'nominal') & (metadata.keep)].index
```

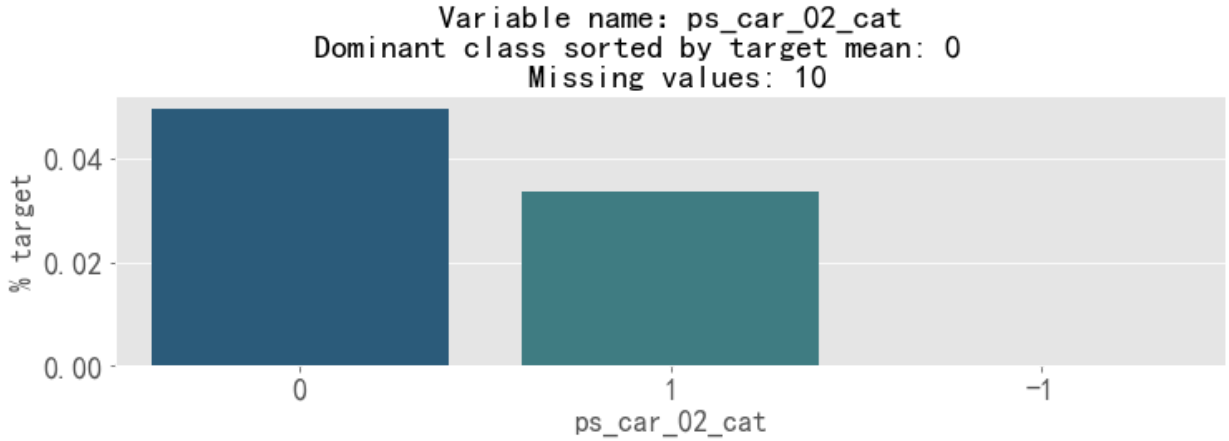
In [9]:
```
for col in cat_cols:
    fig, ax = plt.subplots(figsize=(12,3))
    # Calculate the percentage of target=1 per category value
    cat_perc = train[[col, 'target']].groupby([col],as_index=False).mean()
    cat_perc.sort_values(by='target', ascending=False, inplace=True)
    sns.barplot(ax=ax, x=col, y='target', data=cat_perc, order=cat_perc[col
    plt.ylabel('% target', fontsize=18)
    plt.xlabel(col, fontsize=18)
    plt.tick_params(axis='both', which='major', labelsize=18)
    plt.title(f"Variable name: {col}\n Dominant class sorted by target mean
    \n Missing values: {metadata.loc[col]['missing']}", fontsize = 20)
    plt.show();
```

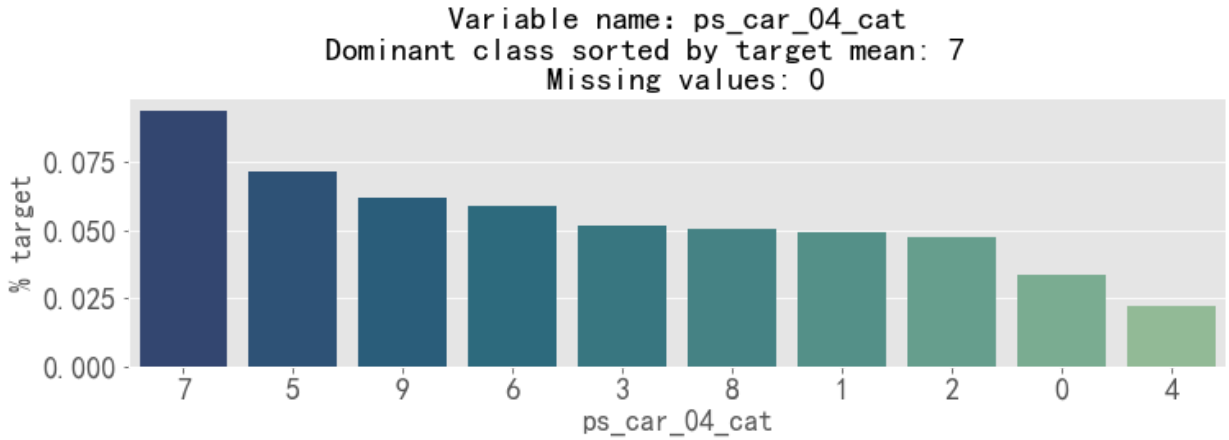


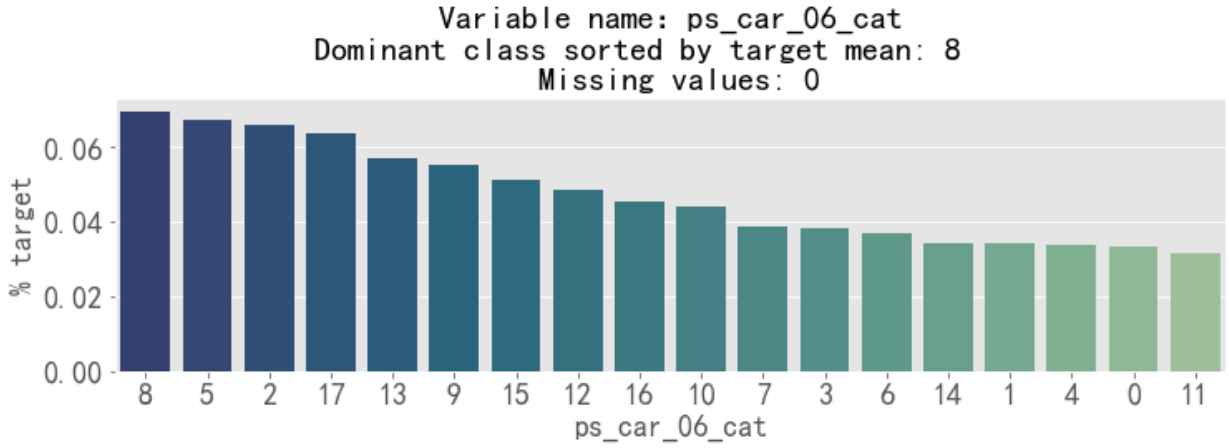


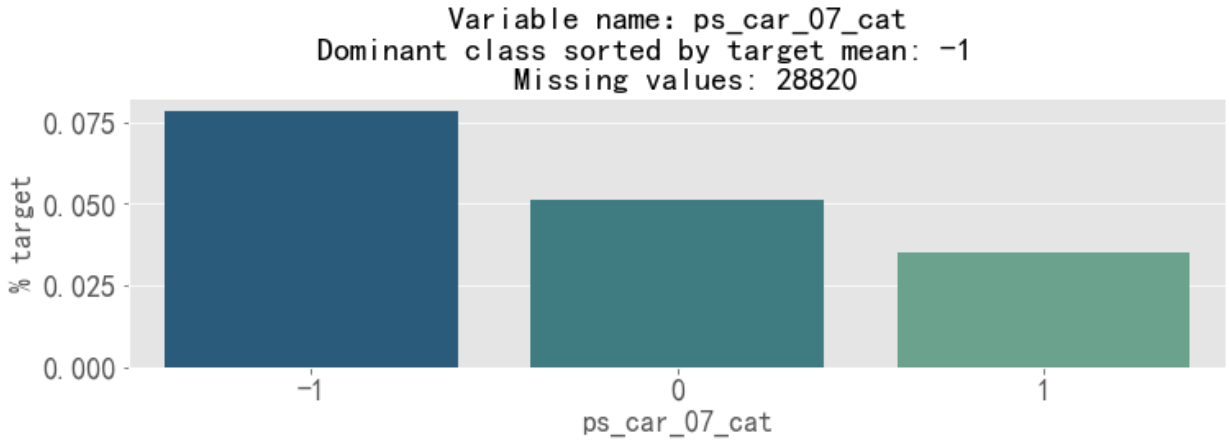


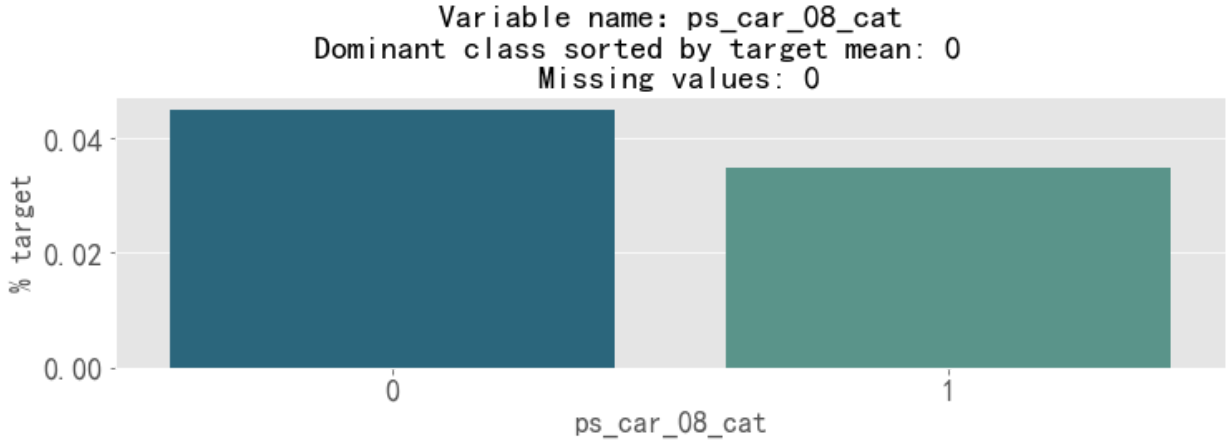

Variable name: ps_car_01_cat
Dominant class sorted by target mean: -1
Missing values: 267
% target
0.3
0.2
0.1
0.0
-1 9 1 0 8 2 11 5 4 3 10 6 7
ps_car_01_cat

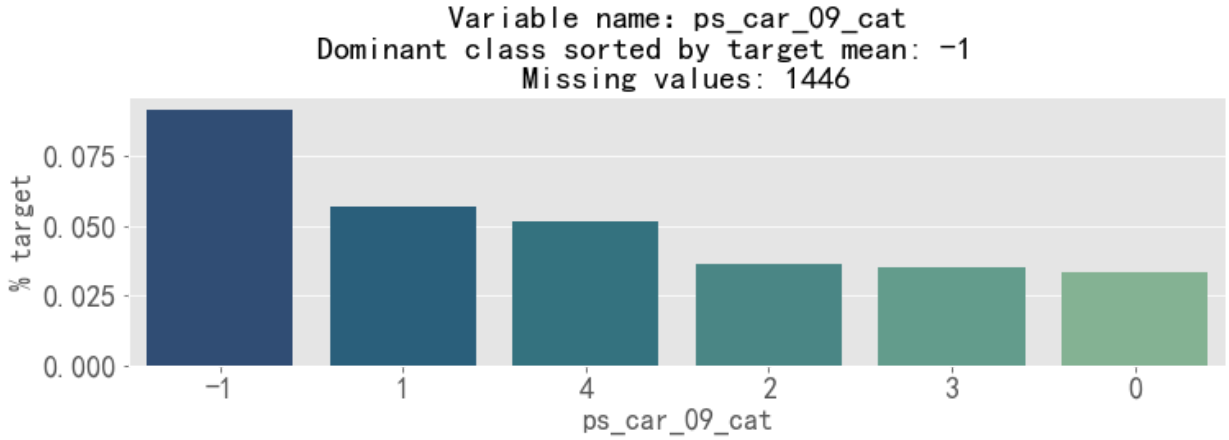

Variable name: ps_car_02_cat
Dominant class sorted by target mean: 0
Missing values: 10
% target
0.04
0.02
0.00
0 1 -1
ps_car_02_cat

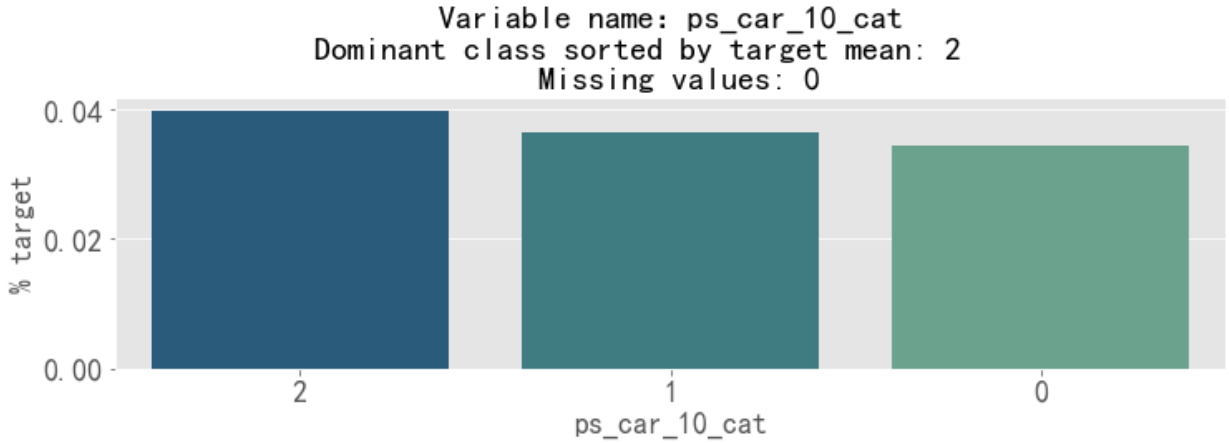

Variable name: ps_car_04_cat
Dominant class sorted by target mean: 7
Missing values: 0
% target
0.075
0.050
0.025
0.000
7 5 9 6 3 8 1 2 0 4
ps_car_04_cat

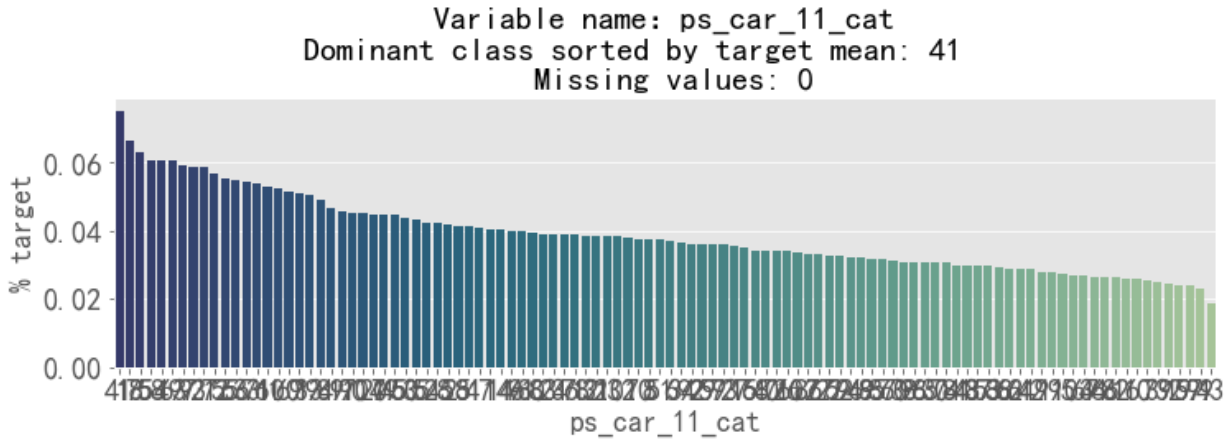

Variable name: ps_car_06_cat
Dominant class sorted by target mean: 8
Missing values: 0
% target
0.06
0.04
0.02
0.00
8 5 2 17 13 9 15 12 16 10 7 3 6 14 1 4 0 11
ps_car_06_cat


Variable name: ps_car_07_cat
Dominant class sorted by target mean: -1
Missing values: 28820
0.075
0.050
0.025
0.000
% target
-1
0
1
ps_car_07_cat

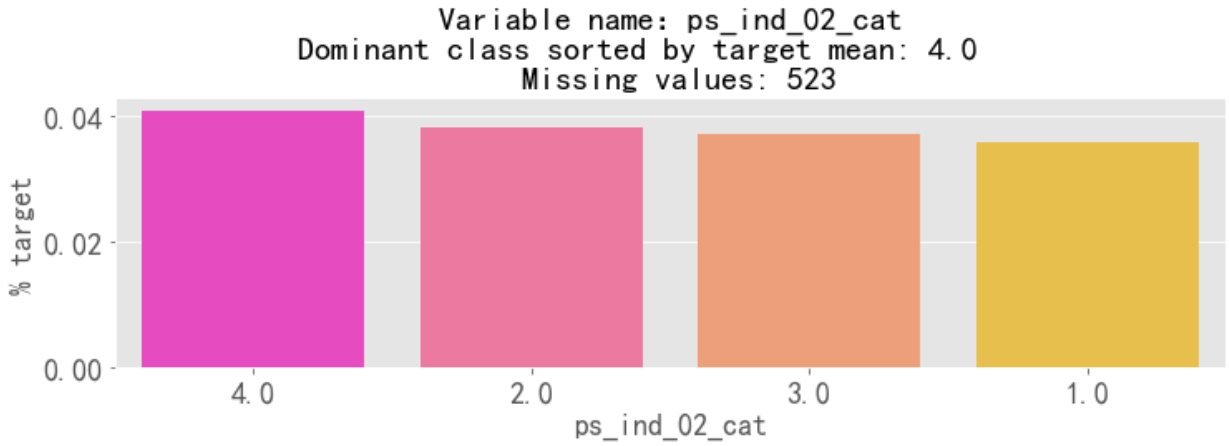

Variable name: ps_car_08_cat
Dominant class sorted by target mean: 0
Missing values: 0
0.04
0.02
0.00
% target
0
1
ps_car_08_cat

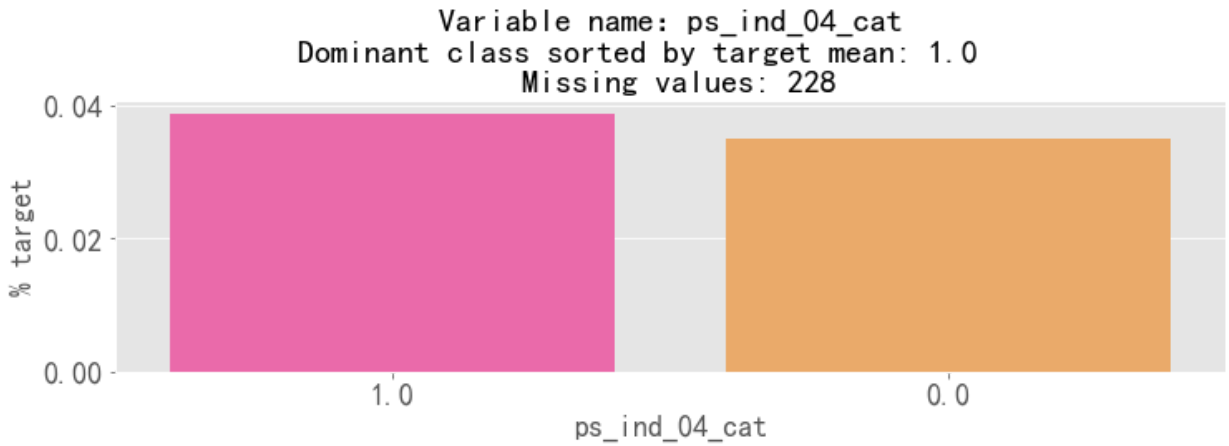

Variable name: ps_car_09_cat
Dominant class sorted by target mean: -1
Missing values: 1446
0.075
0.050
0.025
0.000
% target
-1
1
4
2
3
0
ps_car_09_cat

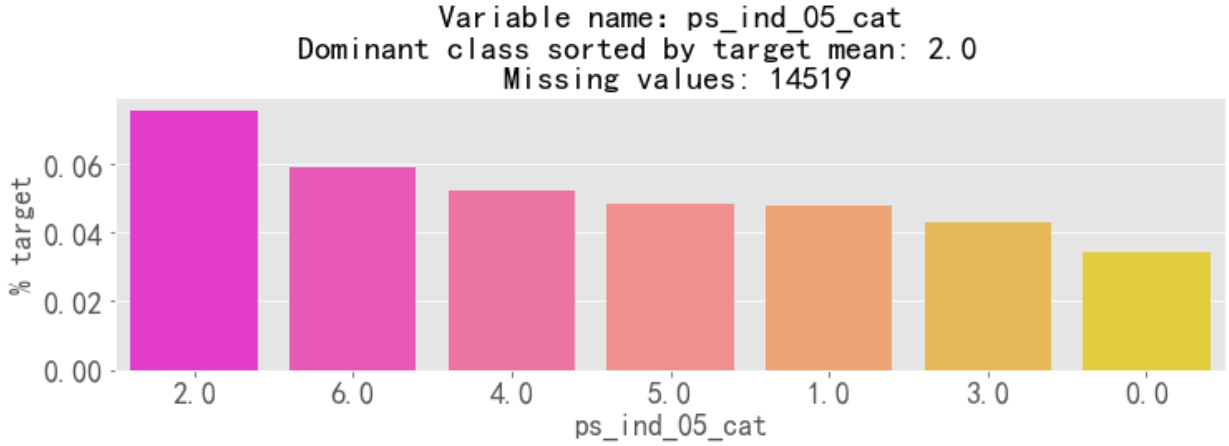

Variable name: ps_car_10_cat
Dominant class sorted by target mean: 2
Missing values: 0
0.04
0.02
0.00
% target
2
1
0
ps_car_10_cat

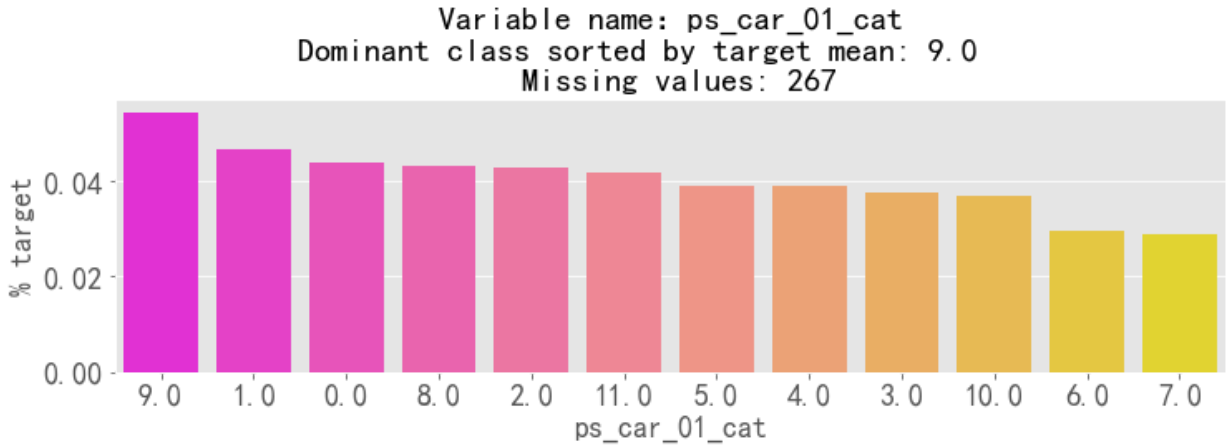

Variable name: ps_car_11_cat
Dominant class sorted by target mean: 41
Missing values: 0
% target
0.06
0.04
0.02
0.00
ps_car_11_cat

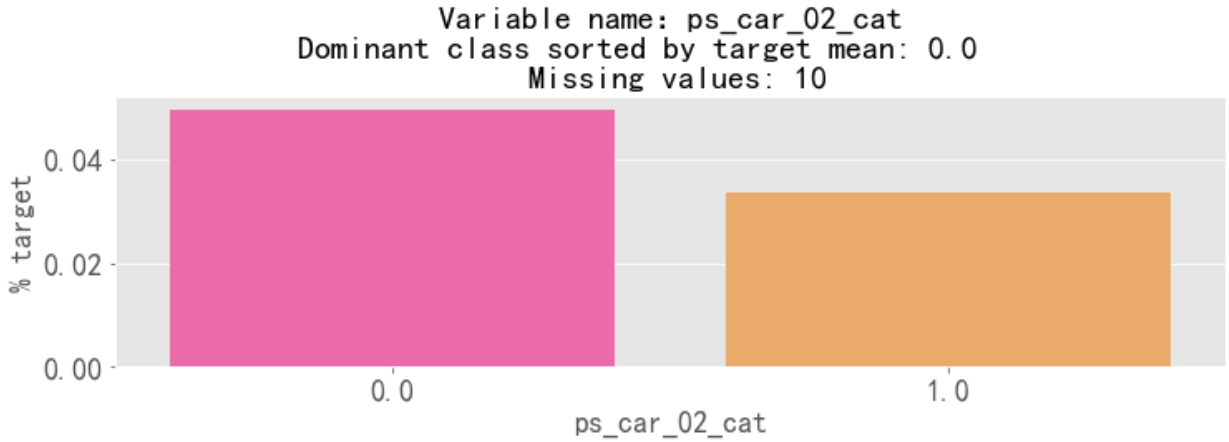
In [10]:
```
for col in cat_cols:
    fig, ax = plt.subplots(figsize=(12,3))
    # Calculate the percentage of target=1 per category value
    cat_perc = train_imp[[col, 'target']].groupby([col],as_index=False).mea
    cat_perc.sort_values(by='target', ascending=False, inplace=True)
    sns.barplot(ax=ax, x=col, y='target', data=cat_perc, order=cat_perc[col
    plt.ylabel('% target', fontsize=18)
    plt.xlabel(col, fontsize=18)
    plt.tick_params(axis='both', which='major', labelsize=18)
    plt.title(f"Variable name: {col}\n Dominant class sorted by target mean
    \n Missing values: {metadata.loc[col]['missing']}", fontsize = 20)
    plt.show();
```

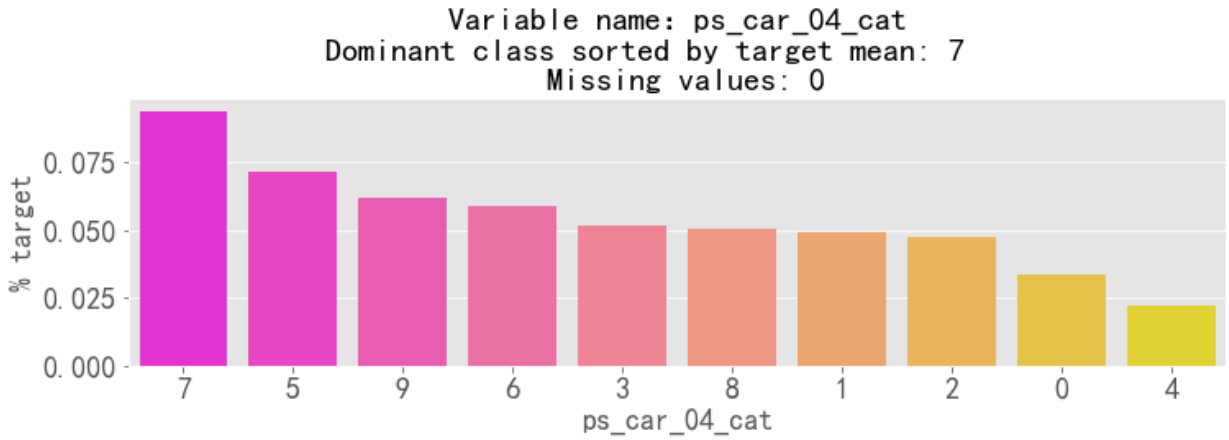


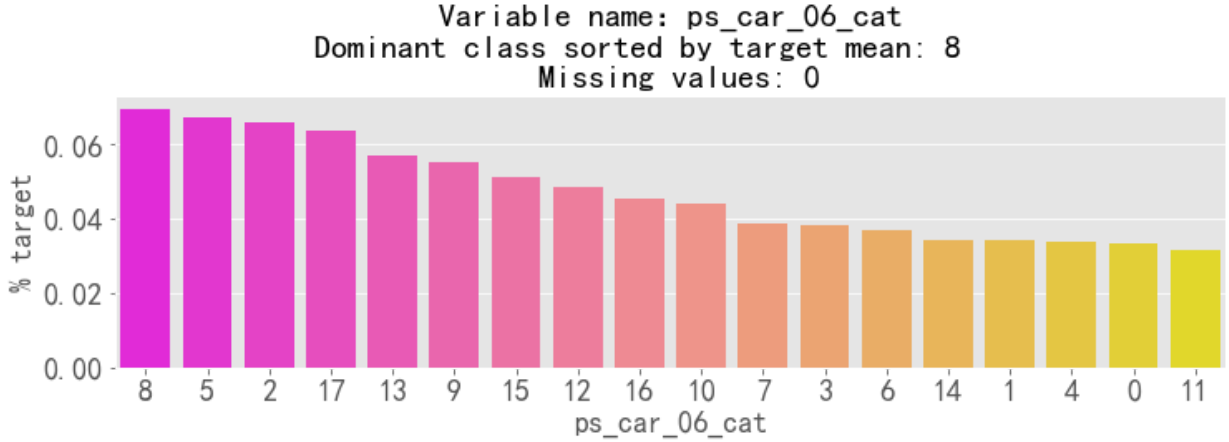


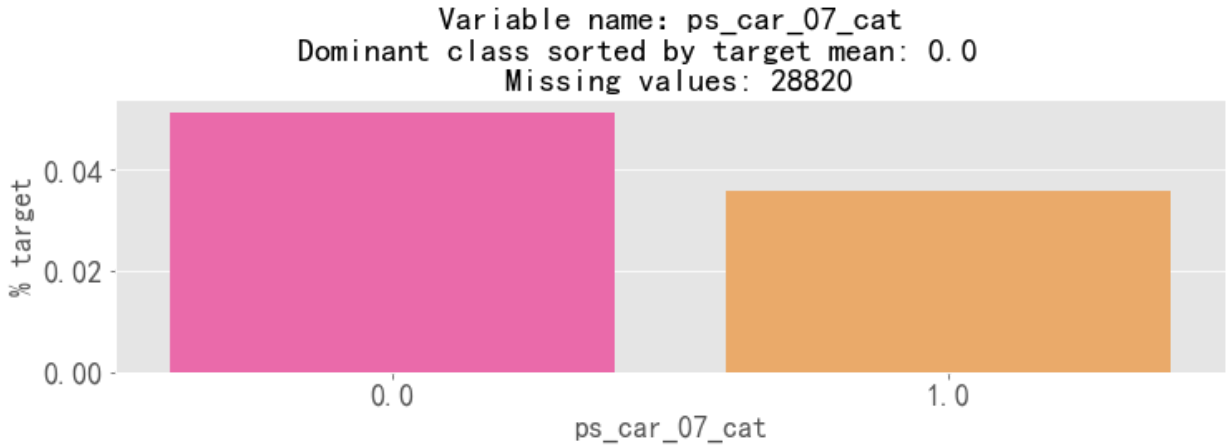


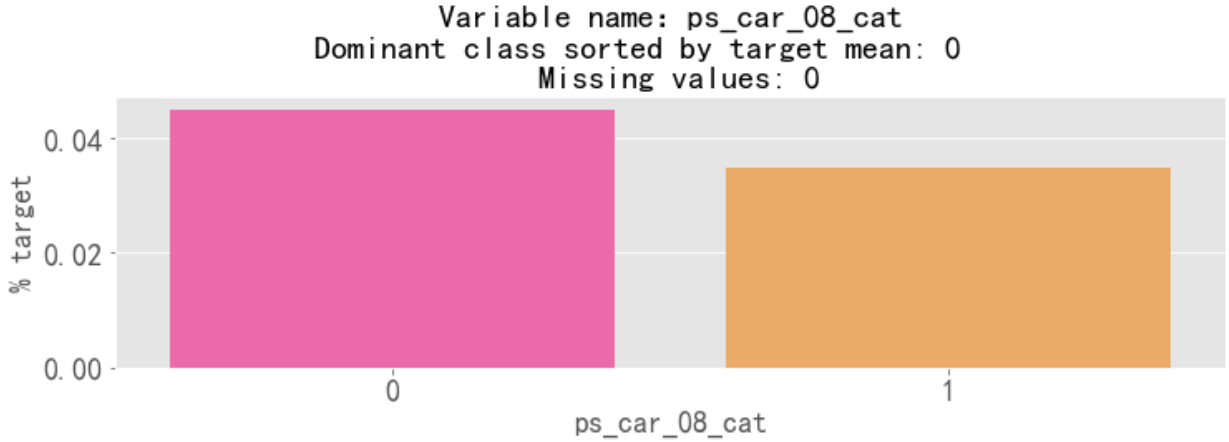

Variable name: ps_car_01_cat
Dominant class sorted by target mean: 9.0
Missing values: 267
% target
0.00
0.02
0.04
9.0
1.0
0.0
8.0
2.0
11.0
5.0
4.0
3.0
10.0
6.0
7.0
ps_car_01_cat

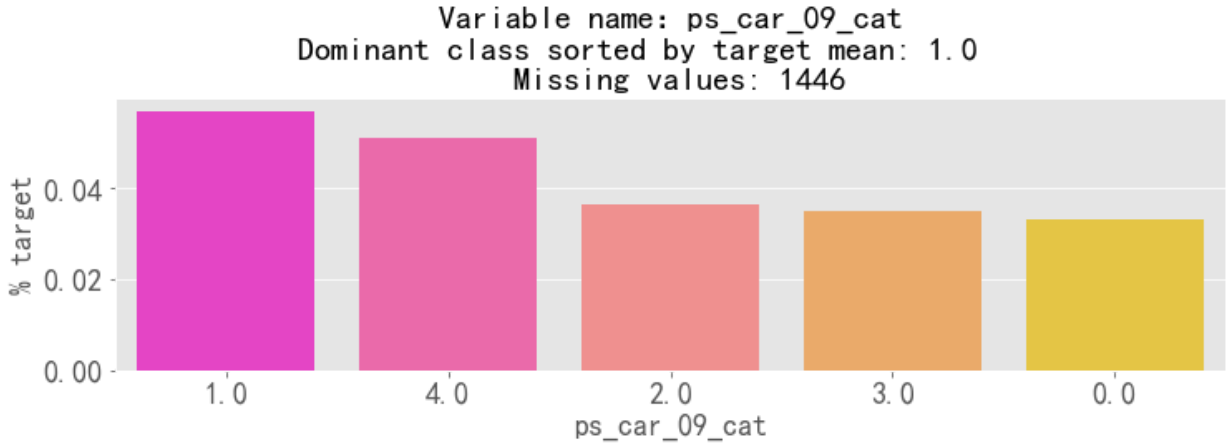

Variable name: ps_car_02_cat
Dominant class sorted by target mean: 0.0
Missing values: 10
% target
0.00
0.02
0.04
0.0
1.0
ps_car_02_cat

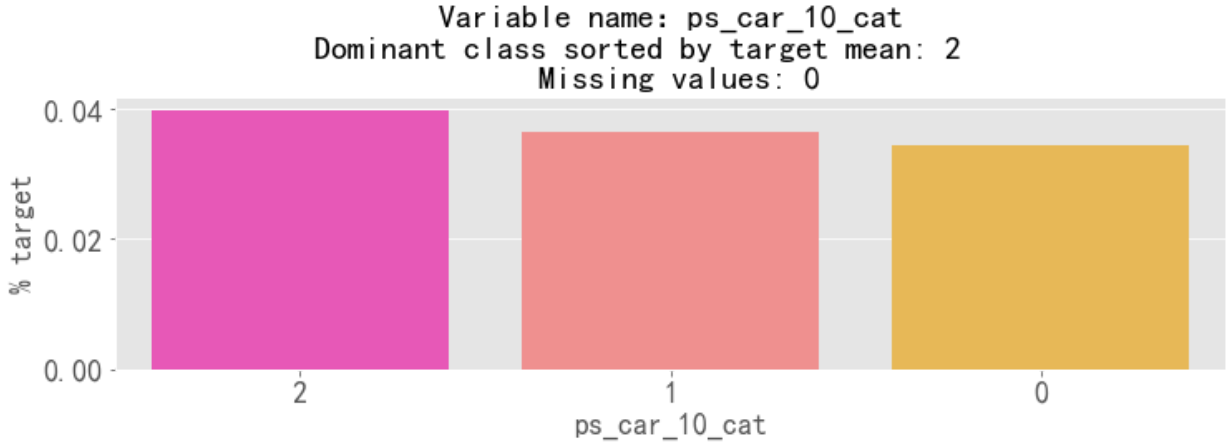

Variable name: ps_car_04_cat
Dominant class sorted by target mean: 7
Missing values: 0
% target
0.000
0.025
0.050
0.075
7
5
9
6
3
8
1
2
0
4
ps_car_04_cat

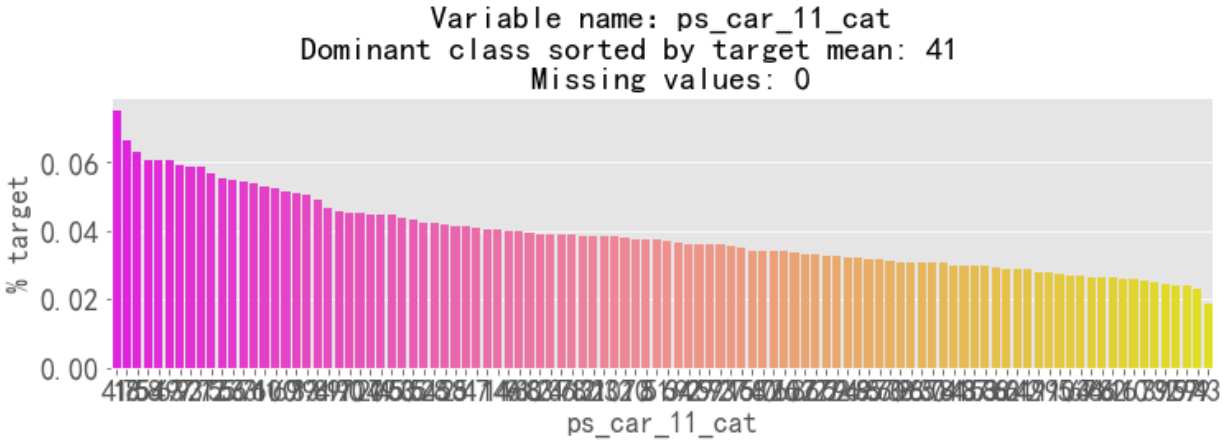

Variable name: ps_car_06_cat
Dominant class sorted by target mean: 8
Missing values: 0
% target
0.00
0.02
0.04
0.06
8
5
2
17
13
9
15
12
16
10
7
3
6
14
1
4
0
11
ps_car_06_cat


Variable name: ps_car_07_cat
Dominant class sorted by target mean: 0.0
Missing values: 28820
% target
0.04
0.02
0.00
0.0
1.0
ps_car_07_cat


Variable name: ps_car_08_cat
Dominant class sorted by target mean: 0
Missing values: 0
% target
0.04
0.02
0.00
0
1
ps_car_08_cat


Variable name: ps_car_09_cat
Dominant class sorted by target mean: 1.0
Missing values: 1446
% target
0.04
0.02
0.00
1.0
4.0
2.0
3.0
0.0
ps_car_09_cat

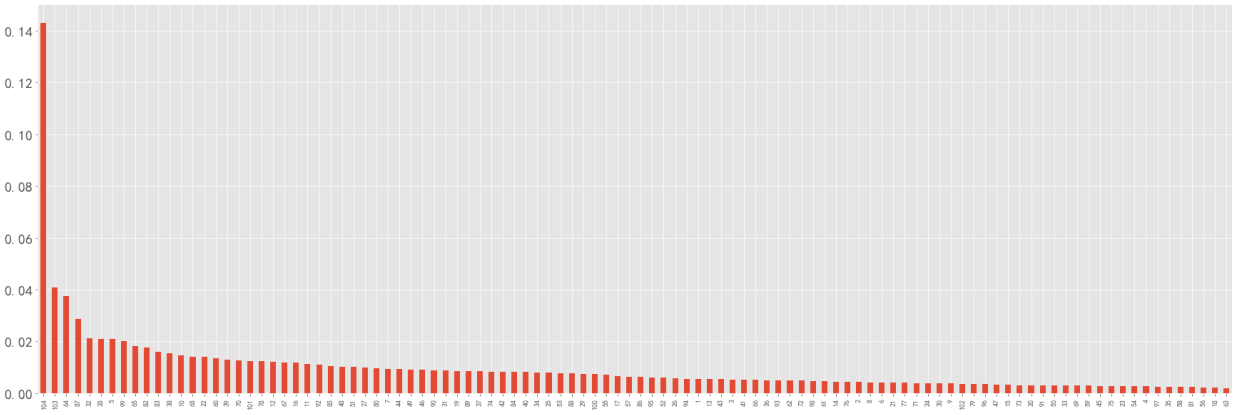

Variable name: ps_car_10_cat
Dominant class sorted by target mean: 2
Missing values: 0
% target
0.04
0.02
0.00
2
1
0
ps_car_10_cat



In [11]:
```
reverse_trans_cols = [
 'ps_car_07_cat',
 'ps_ind_05_cat',
 'ps_car_09_cat',
 'ps_ind_02_cat',
 'ps_car_01_cat',
 'ps_ind_04_cat',
]
```

In [12]:
```
train_imp[reverse_trans_cols] = train[reverse_trans_cols]
```

In [13]:
```
(train_imp.ps_car_11_cat.value_counts()/train_imp.shape[0]).plot(kind='bar'
plt.tick_params(axis='y', which='major', labelsize=20)
```



Binary

In [14]:
```
bin_cols = metadata[(metadata.level == 'binary') & (metadata.keep)].index
```

In [15]:
```
zero_list = []
one_list = []
for col in bin_cols:
    zero_list.append((train[col]==0).sum())
    one_list.append((train[col]==1).sum())
```

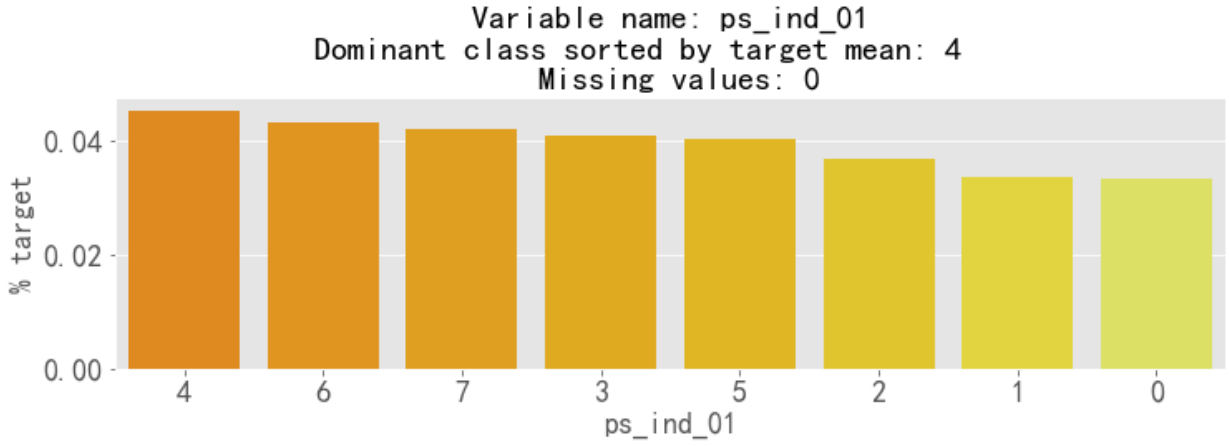
In [16]:
```
trace1 = go.Bar(
    x=bin_cols,
    y=zero_list ,
    name='Zero count'
)
trace2 = go.Bar(
    x=bin_cols,
    y=one_list,
    name='One count'
)

data = [trace1, trace2]
layout = go.Layout(
    barmode='stack',
    title='Count of 1 and 0 in binary variables including TARGET'
)

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='stacked-bar')
```

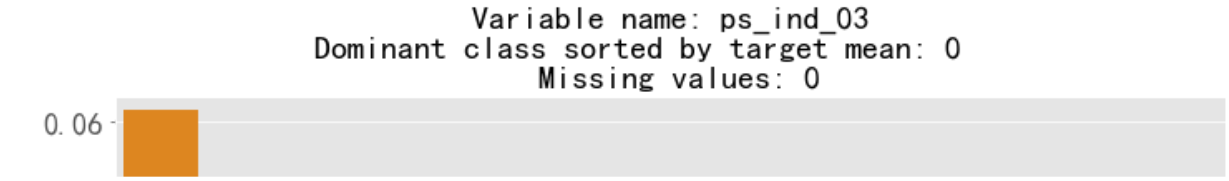
Here we observe that there are 4 features : ps_ind_10_bin, ps_ind_11_bin, ps_ind_12_bin, ps_ind_13_bin which are completely dominated by zeros. This begs the question of whether these features are useful at all as they do not contain much information about the other class vis-a-vis the target.

Ordinal

In [17]:
```
ord_cols = metadata[(metadata.level == 'ordinal') & (metadata.keep)].index
```

In [18]:
```
for col in ord_cols:
    fig, ax = plt.subplots(figsize=(12,3))
    # Calculate the percentage of target=1 per category value
    cat_perc = train[[col, 'target']].groupby([col],as_index=False).mean()
    cat_perc.sort_values(by='target', ascending=False, inplace=True)
    sns.barplot(ax=ax, x=col, y='target', data=cat_perc, order=cat_perc[col
    plt.ylabel('% target', fontsize=18)
    plt.xlabel(col, fontsize=18)
    plt.tick_params(axis='both', which='major', labelsize=18)
    plt.title(f"Variable name: {col}\n Dominant class sorted by target mean
    \n Missing values: {metadata.loc[col]['missing']}", fontsize = 20)
    plt.show();
```



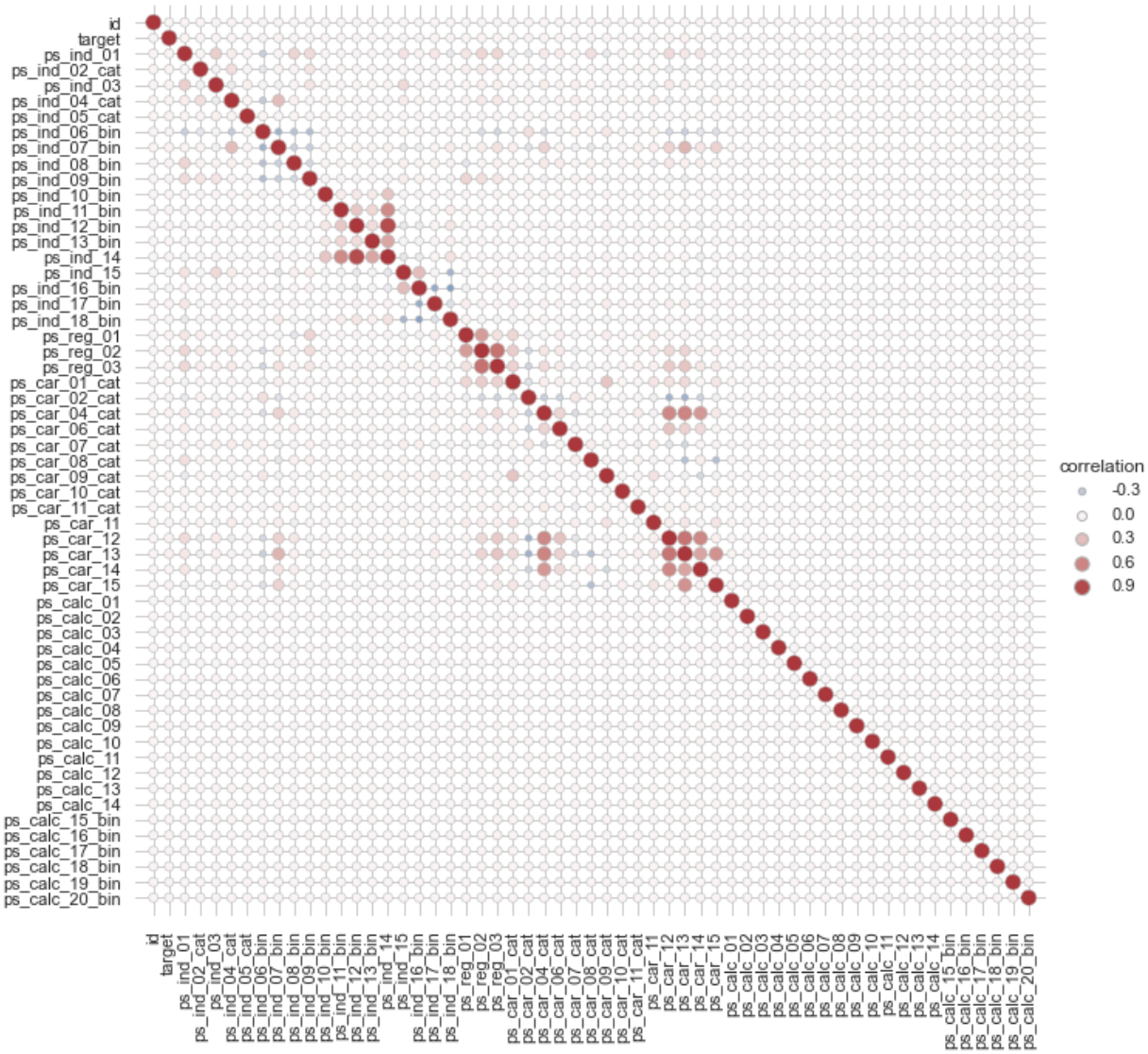


In [19]:
```
conti_cols = metadata[(metadata.level.isin(['ratio','interval'])) & (metada
```

In [20]:
```
conti_cols
```

Out[20]:
```
Index(['ps_reg_01', 'ps_reg_02', 'ps_reg_03', 'ps_car_12', 'ps_car_13',
       'ps_car_14', 'ps_car_15', 'ps_calc_01', 'ps_calc_02', 'ps_calc_0
3'],
      dtype='object', name='colname')
```

In [21]:
```
sns.set_theme(style="whitegrid")
corr_mat = train_imp.corr().stack().reset_index(name="correlation")

# Draw each cell as a scatter point with varying size and color
g = sns.relplot(
    data=corr_mat,
    x="level_0", y="level_1", hue="correlation", size="correlation",
    palette="vlag", hue_norm=(-1, 1), edgecolor=".7",
    height=10, sizes=(20, 100), size_norm=(-.2, .8),
)

# Tweak the figure to finalize
g.set(xlabel="", ylabel="", aspect="equal")
g.despine(left=True, bottom=True)
g.ax.margins(.02)
for label in g.ax.get_xticklabels():
    label.set_rotation(90)
for artist in g.legend.legendHandles:
    artist.set_edgecolor(".7")
```



In [22]:
```
# Generate a mask for the upper triangle

corr_mat = train_imp[conti_cols].corr()
mask = np.triu(np.ones_like(corr_mat, dtype=bool))

# Set up the matplotlib figure
f, ax = plt.subplots(figsize=(11, 9))

# Generate a custom diverging colormap
cmap = sns.diverging_palette(230, -20, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr_mat, mask=mask, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .75},annot=True
```

Out[22]: <AxesSubplot:>

There are a strong correlations between the variables:

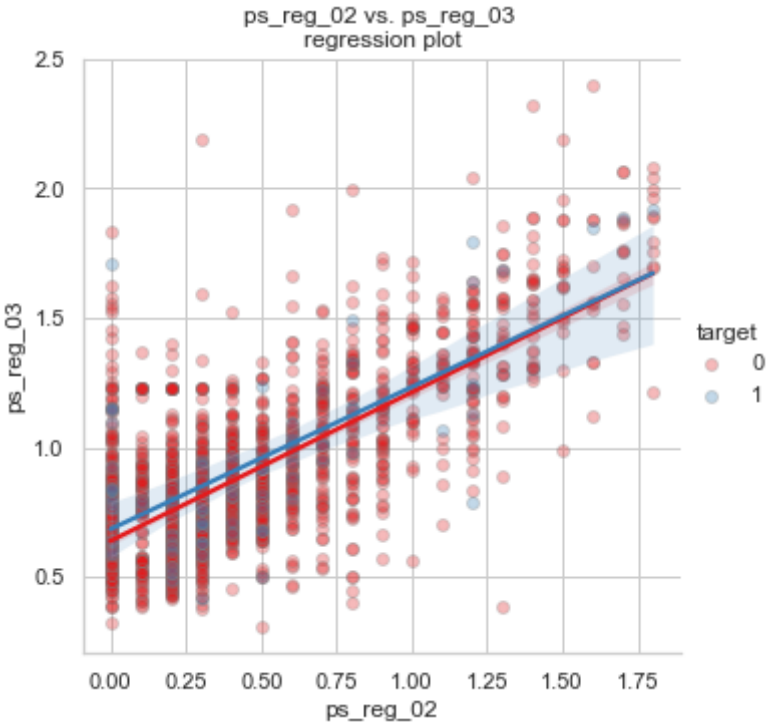
- ps_reg_02 and ps_reg_03 (0.69)
- ps_car_12 and ps_car_13 (0.67)
- ps_car_12 and ps_car_14 (0.58)
- ps_car_13 and ps_car_15 (0.53)

- ps_reg_01 and ps_reg_02 (0.47)

In [23]:
```
trn_sample = train_imp.sample(2000)
```

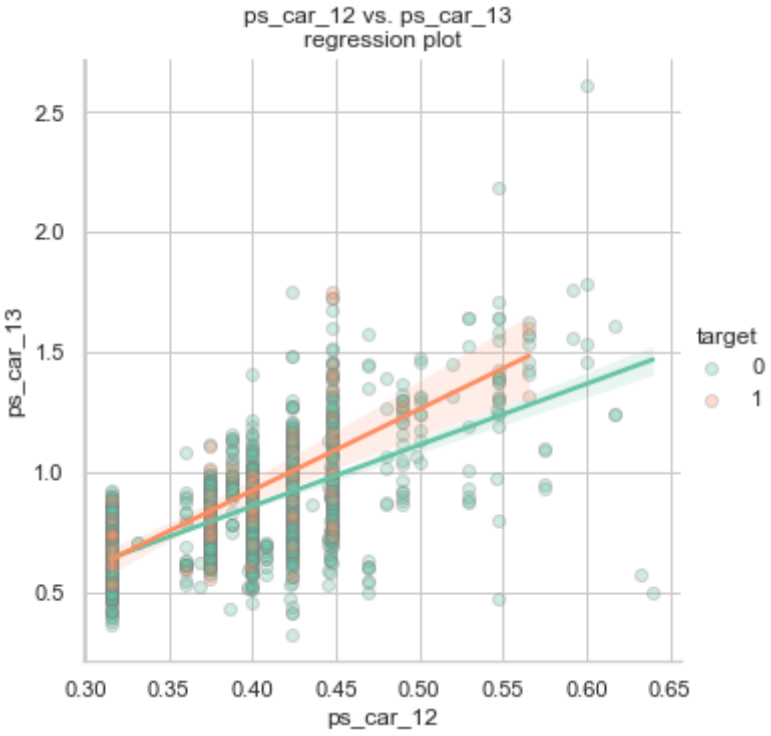
In [24]:
```
sns.lmplot(x='ps_reg_02', y='ps_reg_03', data=trn_sample, hue='target',
           palette='Set1', height = 5, scatter_kws={'alpha':0.3,'edgecolor'
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x19f00194dc8>
```



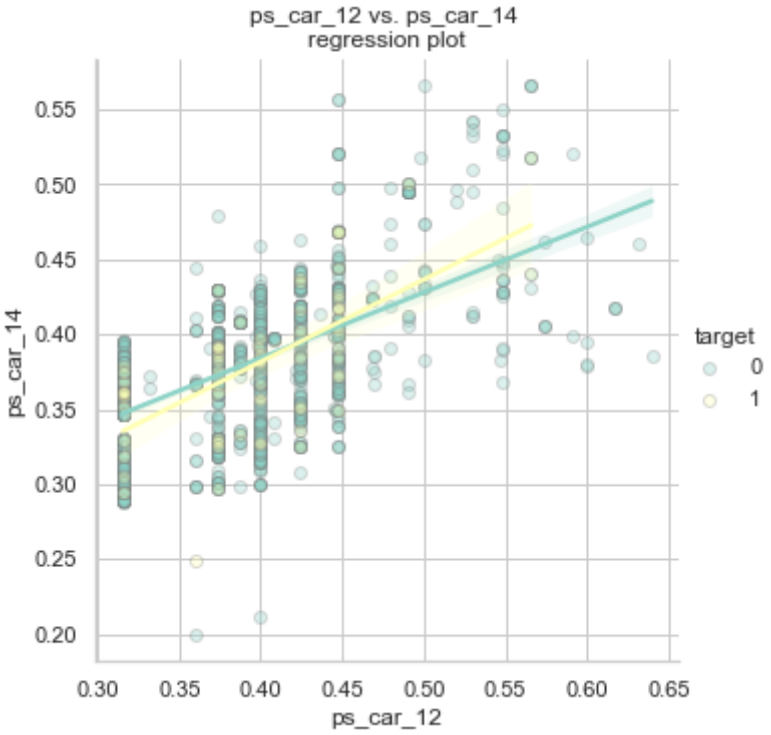
In [25]:
```
sns.lmplot(x='ps_car_12', y='ps_car_13', data=trn_sample, hue='target',
           palette='Set2', height = 5, scatter_kws={'alpha':0.3,'edgecolor'
```

Out[25]: <seaborn.axisgrid.FacetGrid at 0x19f0fda9108>



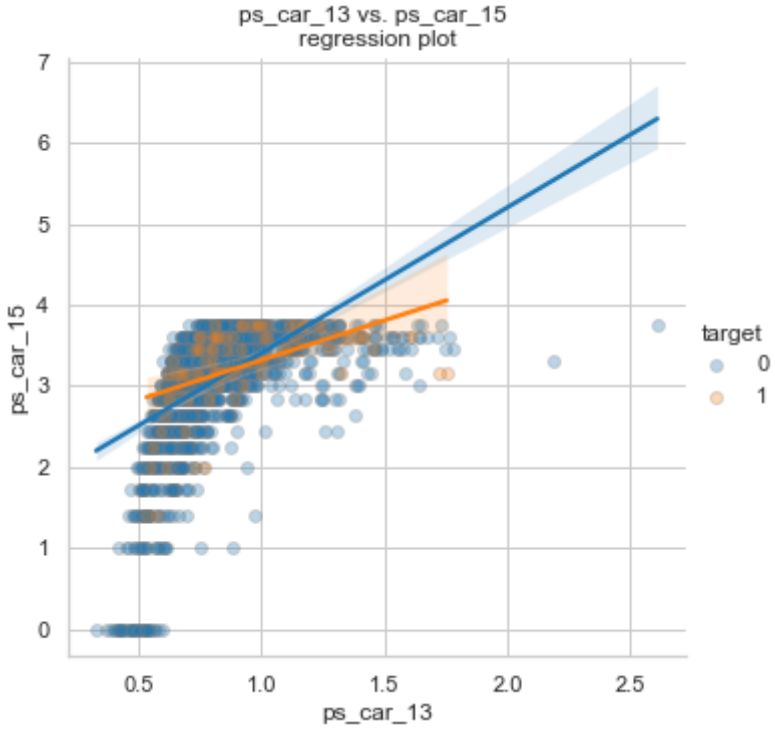
In [26]:
```
sns.lmplot(x='ps_car_12', y='ps_car_14', data=trn_sample, hue='target',
           palette='Set3', height = 5, scatter_kws={'alpha':0.3,'edgecolor'
```

Out[26]: <seaborn.axisgrid.FacetGrid at 0x19f003da548>



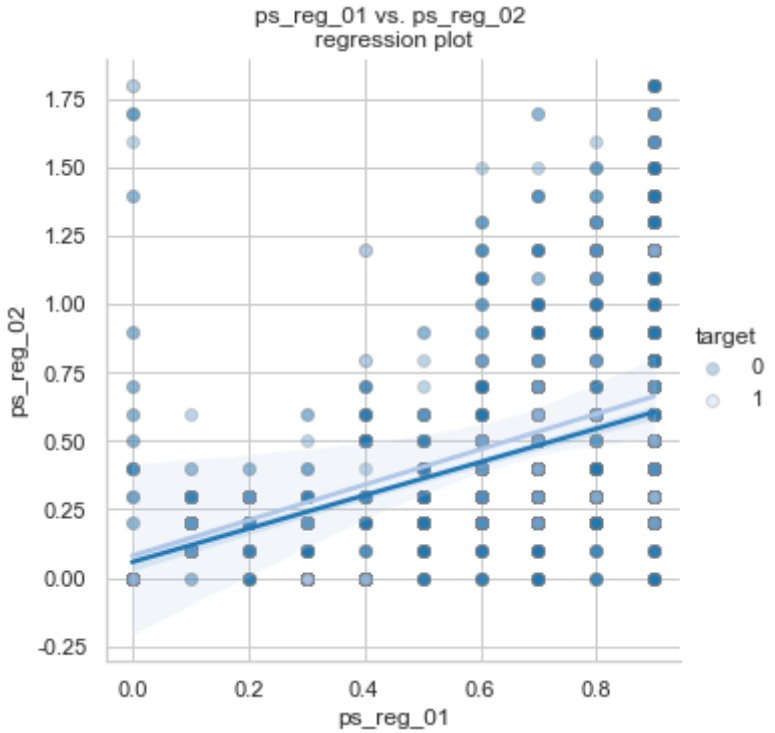
In [27]:
```
sns.lmplot(x='ps_car_13', y='ps_car_15', data=trn_sample, hue='target',
           palette='tab10', height = 5, scatter_kws={'alpha':0.3,'edgecolor
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x19f0f8fa8c8>



In [28]:
```
sns.lmplot(x='ps_reg_01', y='ps_reg_02', data=trn_sample, hue='target',
           palette='tab20', height = 5, scatter_kws={'alpha':0.3,'edgecolor
```

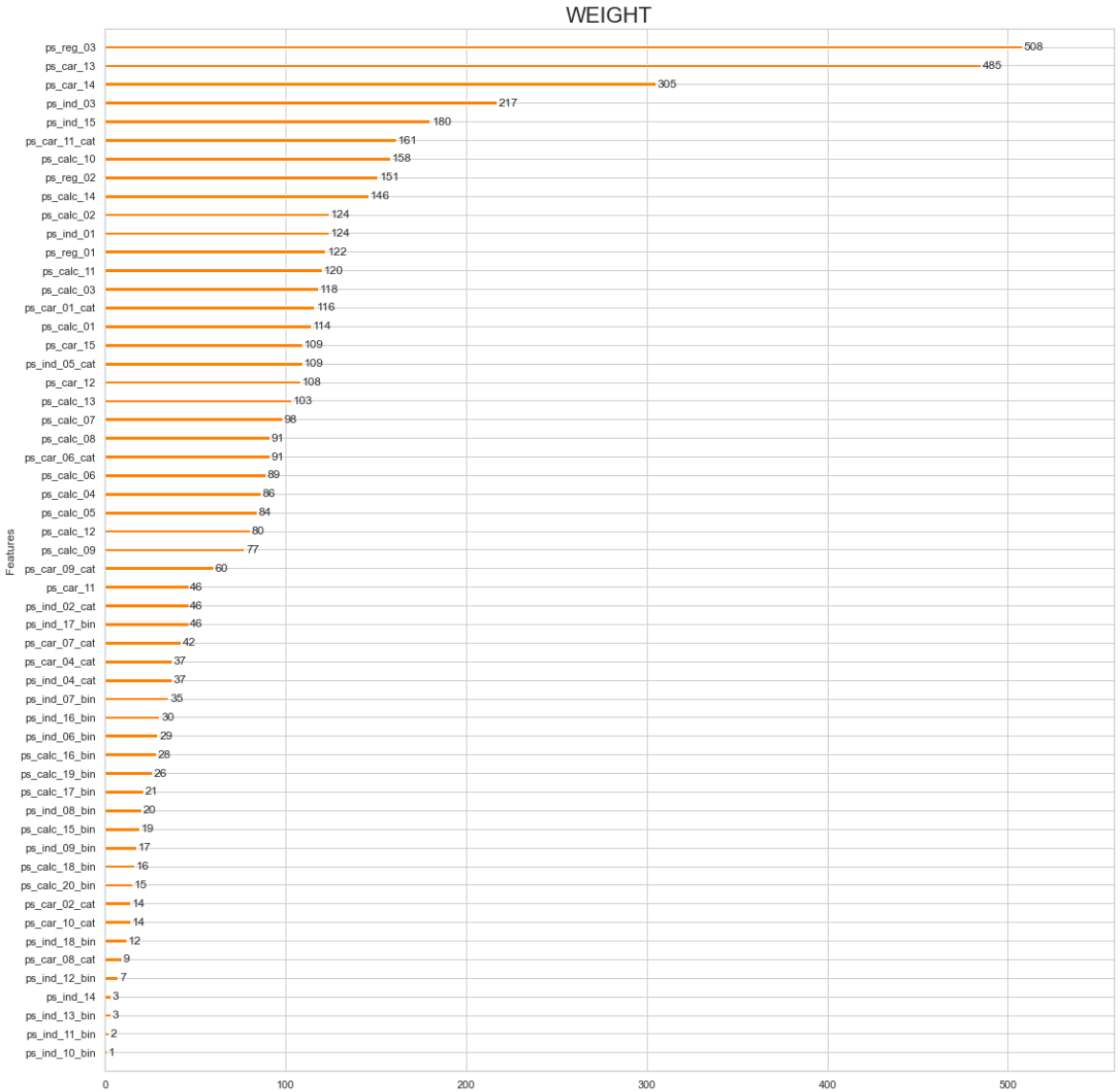
Out[28]: <seaborn.axisgrid.FacetGrid at 0x19f0fe41888>

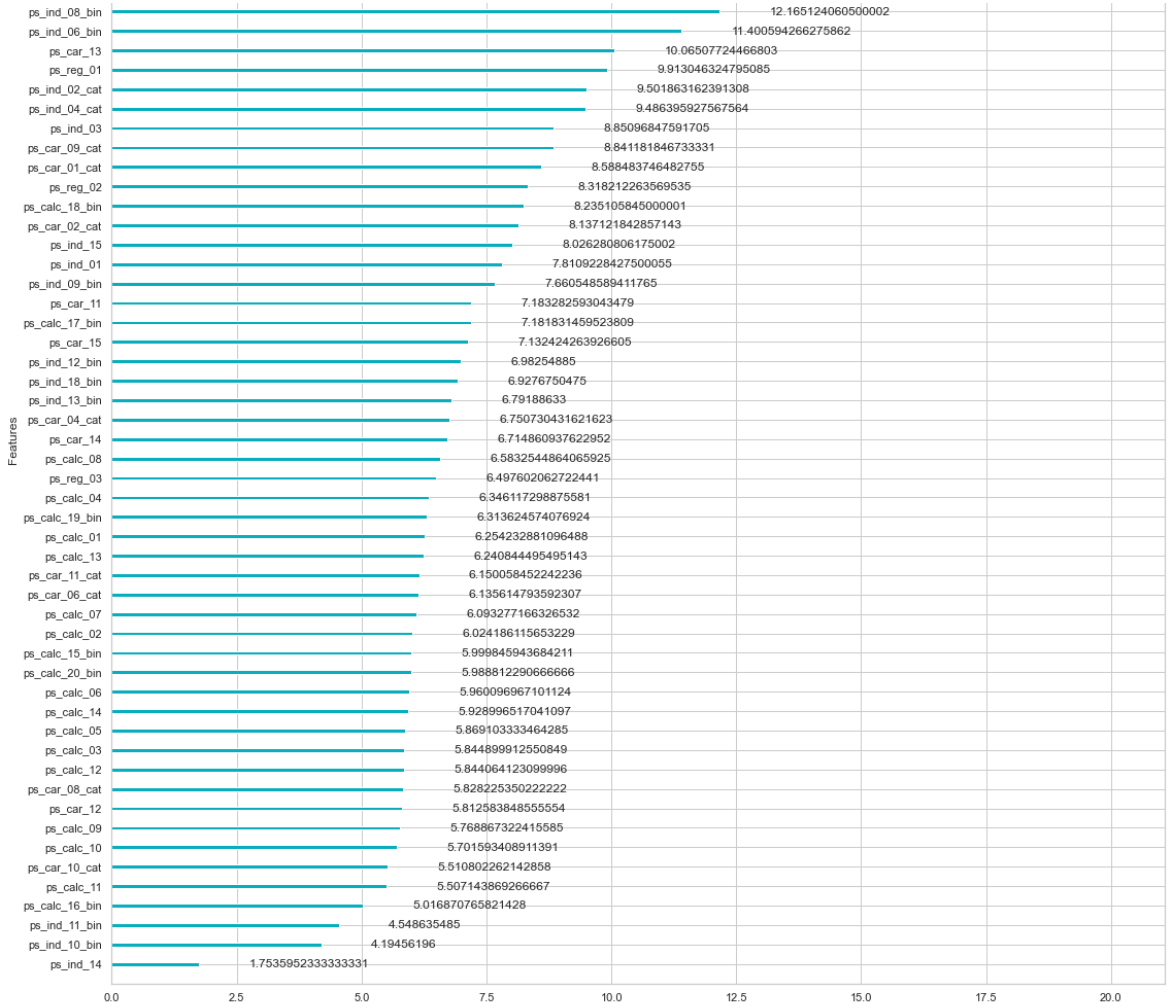


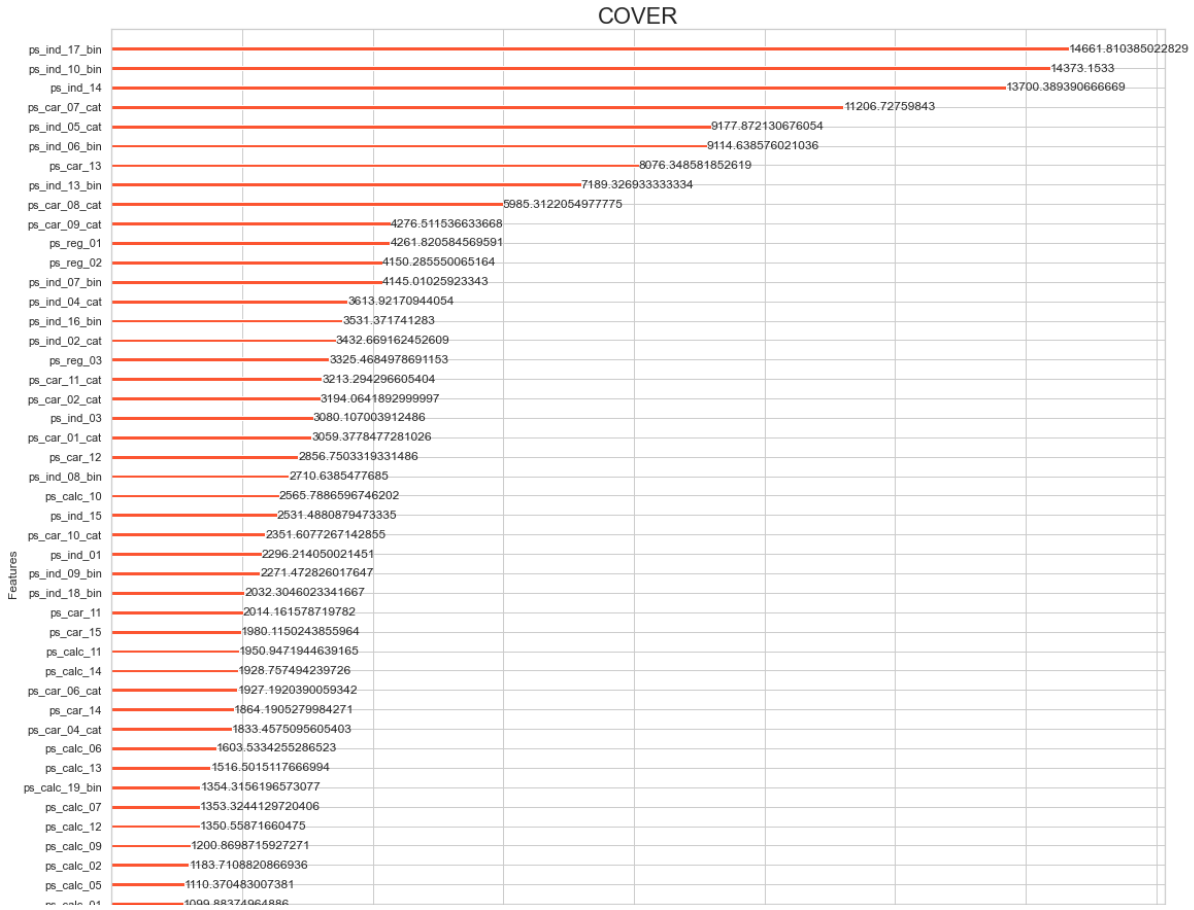
In [29]:
```
from xgboost import XGBClassifier
from xgboost import plot_importance
plt.figure(figsize = [100,20])

X = train_imp.drop(['id', 'target'], axis=1)
y = train_imp.target

model = XGBClassifier()

model.fit(X, y)
# plot feature importance
```

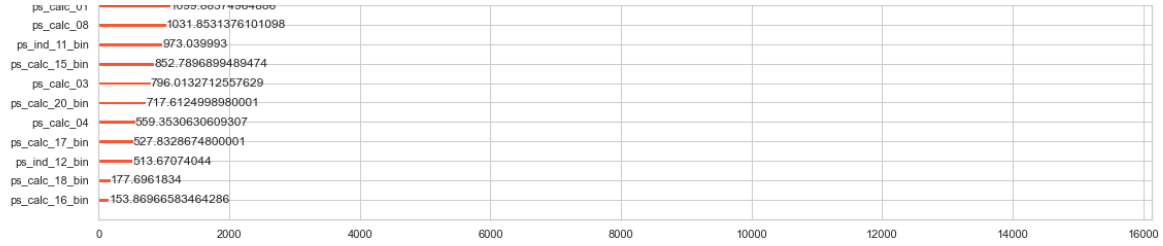
Out[29]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_sta
te=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)

<Figure size 7200x1440 with 0 Axes>
```

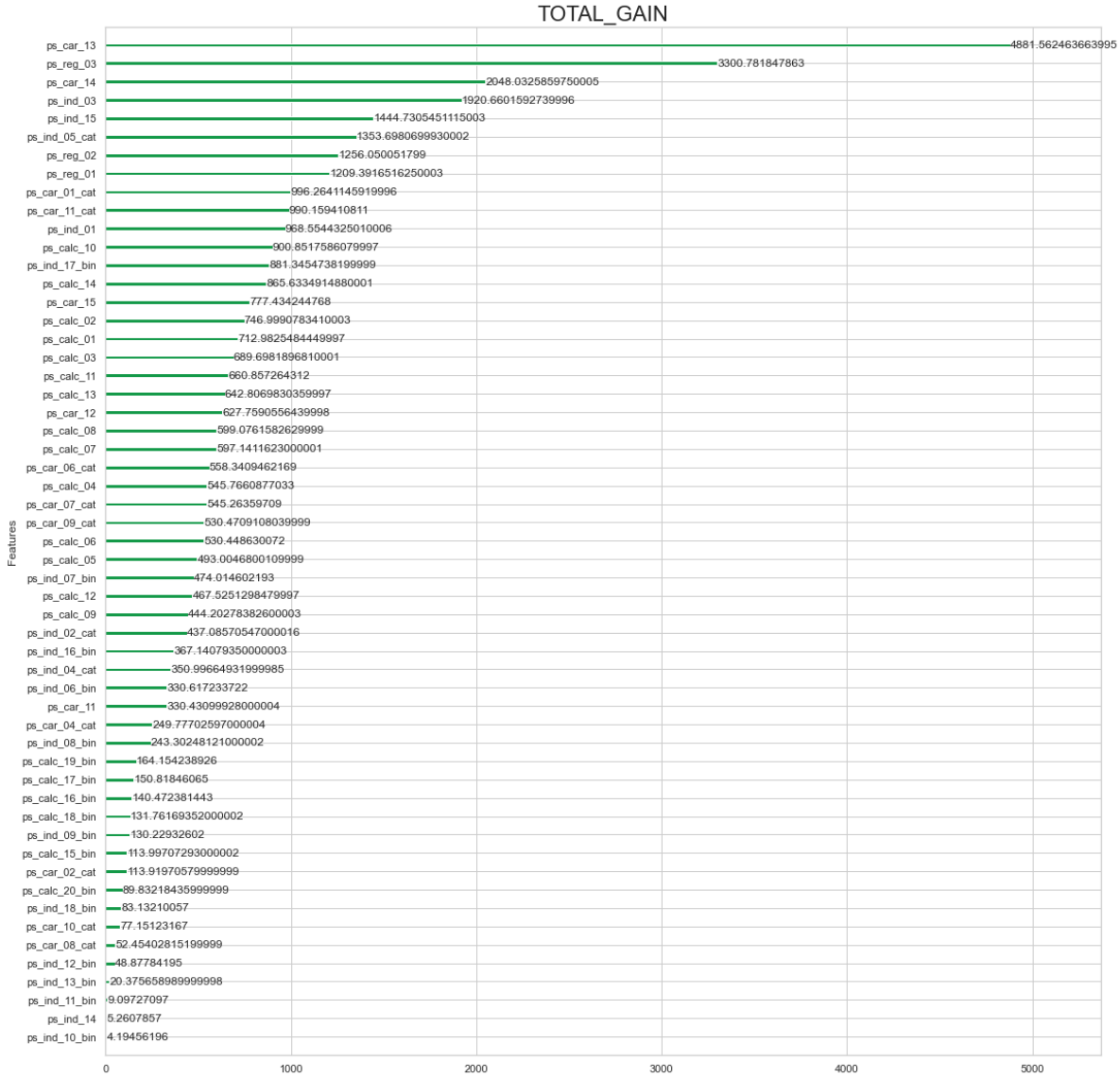
In [30]:
```
# define subplot grid
fig, axs = plt.subplots(nrows=5, ncols=1, figsize=(15, 80))
plt.tight_layout()
plt.subplots_adjust(hspace=0.1)

types = ['weight', 'gain', 'cover', 'total_gain', 'total_cover']
# loop through tickers and axes
colors = ['#ff7f01','#08aebd','#fc5531','#139948','#8950fe']
for ty, ax, color in zip(types, axs.ravel(), colors):
    # filter df for ticker and plot on specified axes
    plot_importance(ax = ax, booster = model,importance_type=ty, color = co

    # chart formatting
    ax.set_title(ty.upper(),fontsize = 22)
    ax.set_xlabel("")

plt.show()
```




ps_ind_08_bin 12.165124060500002
ps_ind_06_bin 11.400594266275862
ps_car_13 10.06507724466803
ps_reg_01 9.913046324795085
ps_ind_02_cat 9.501863162391308
ps_ind_04_cat 9.486395927567564
ps_ind_03 8.85096847591705
ps_car_09_cat 8.841181846733331
ps_car_01_cat 8.588483746482755
ps_reg_02 8.318212263569535
ps_calc_18_bin 8.235105845000001
ps_car_02_cat 8.137121842857143
ps_ind_15 8.026280806175002
ps_ind_01 7.8109228427500055
ps_ind_09_bin 7.660548589411765
ps_car_11 7.183282593043479
ps_calc_17_bin 7.181831459523809
ps_car_15 7.132424263926605
ps_ind_12_bin 6.98254885
ps_ind_18_bin 6.9276750475
ps_ind_13_bin 6.79188633
ps_car_04_cat 6.750730431621623
ps_car_14 6.714860937622952
ps_calc_08 6.5832544864065925
ps_reg_03 6.497602062722441
ps_calc_04 6.346117298875581
ps_calc_19_bin 6.313624574076924
ps_calc_01 6.254232881096488
ps_calc_13 6.240844495495143
ps_car_11_cat 6.150058452242236
ps_car_06_cat 6.135614793592307
ps_calc_07 6.093277166326532
ps_calc_02 6.024186115653229
ps_calc_15_bin 5.999845943684211
ps_calc_20_bin 5.988812290666666
ps_calc_06 5.960096967101124
ps_calc_14 5.928996517041097
ps_calc_05 5.869103333464285
ps_calc_03 5.844899912550849
ps_calc_12 5.844064123099996
ps_car_08_cat 5.828225350222222
ps_car_12 5.812583848555554
ps_calc_09 5.768867322415585
ps_calc_10 5.701593408911391
ps_car_10_cat 5.510802262142858
ps_calc_11 5.507143869266667
ps_calc_16_bin 5.016870765821428
ps_ind_11_bin 4.548635485
ps_ind_10_bin 4.19456196
ps_ind_14 1.7535952333333331
Features
0.0 2.5 5.0 7.5 10.0 12.5 15.0 17.5 20.0


COVER
ps_ind_17_bin 14661.810385022829
ps_ind_10_bin 14373.1533
ps_ind_14 13700.389390666669
ps_car_07_cat 11206.72759843
ps_ind_05_cat 9177.872130676054
ps_ind_06_bin 9114.638576021036
ps_car_13 8076.348581852619
ps_ind_13_bin 7189.326933333334
ps_car_08_cat 5985.3122054977775
ps_car_09_cat 4276.511536633668
ps_reg_01 4261.820584569591
ps_reg_02 4150.285550065164
ps_ind_07_bin 4145.01025923343
ps_ind_04_cat 3613.92170944054
ps_ind_16_bin 3531.371741283
ps_ind_02_cat 3432.669162452609
ps_reg_03 3325.4684978691153
ps_car_11_cat 3213.294296605404
ps_car_02_cat 3194.0641892999997
ps_ind_03 3080.107003912486
ps_car_01_cat 3059.3778477281026
ps_car_12 2856.7503319331486
ps_ind_08_bin 2710.6385477685
ps_calc_10 2565.7886596746202
ps_ind_15 2531.4880879473335
ps_car_10_cat 2351.6077267142855
ps_ind_01 2296.214050021451
ps_ind_09_bin 2271.472826017647
ps_ind_18_bin 2032.3046023341667
ps_car_11 2014.161578719782
ps_car_15 1980.1150243855964
ps_calc_11 1950.9471944639165
ps_calc_14 1928.757494239726
ps_car_06_cat 1927.1920390059342
ps_car_14 1864.1905279984271
ps_car_04_cat 1833.4575095605403
ps_calc_06 1603.5334255286523
ps_calc_13 1516.5015117666994
ps_calc_19_bin 1354.3156196573077
ps_calc_07 1353.3244129720406
ps_calc_12 1350.55871660475
ps_calc_09 1200.8698715927271
ps_calc_02 1183.7108820866936
ps_calc_05 1110.370483007381
Features


ps_calc_08 1031.8531376101098
ps_ind_11_bin 973.039993
ps_calc_15_bin 852.7896899489474
ps_calc_03 796.0132712557629
ps_calc_20_bin 717.6124998980001
ps_calc_04 559.3530630609307
ps_calc_17_bin 527.8328674800001
ps_ind_12_bin 513.67074044
ps_calc_18_bin 177.6961834
ps_calc_16_bin 153.86966583464286
0 2000 4000 6000 8000 10000 12000 14000 16000


TOTAL_GAIN
Features
ps_car_13 4881.562463663995
ps_reg_03 3300.781847863
ps_car_14 2048.0325859750005
ps_ind_03 1920.6601592739996
ps_ind_15 1444.7305451115003
ps_ind_05_cat 1353.6980699930002
ps_reg_02 1256.050051799
ps_reg_01 1209.3916516250003
ps_car_01_cat 996.2641145919996
ps_car_11_cat 990.159410811
ps_ind_01 968.5544325010006
ps_calc_10 900.8517586079997
ps_ind_17_bin 881.3454738199999
ps_calc_14 865.6334914880001
ps_car_15 777.434244768
ps_calc_02 746.9990783410003
ps_calc_01 712.9825484449997
ps_calc_03 689.6981896810001
ps_calc_11 660.857264312
ps_calc_13 642.8069830359997
ps_car_12 627.7590556439998
ps_calc_08 599.0761582629999
ps_calc_07 597.1411623000001
ps_car_06_cat 558.3409462169
ps_calc_04 545.7660877033
ps_car_07_cat 545.26359709
ps_car_09_cat 530.4709108039999
ps_calc_06 530.448630072
ps_calc_05 493.0046800109999
ps_ind_07_bin 474.014602193
ps_calc_12 467.5251298479997
ps_calc_09 444.20278382600003
ps_ind_02_cat 437.08570547000016
ps_ind_16_bin 367.14079350000003
ps_ind_04_cat 350.99664931999985
ps_ind_06_bin 330.617233722
ps_car_11 330.43099928000004
ps_car_04_cat 249.77702597000004
ps_ind_08_bin 243.30248121000002
ps_calc_19_bin 164.154238926
ps_calc_17_bin 150.81846065
ps_calc_16_bin 140.472381443
ps_calc_18_bin 131.76169352000002
ps_ind_09_bin 130.22932602
ps_calc_15_bin 113.99707293000002
ps_car_02_cat 113.91970579999999
ps_calc_20_bin 89.83218435999999
ps_ind_18_bin 83.13210057
ps_car_10_cat 77.15123167
ps_car_08_cat 52.45402815199999
ps_ind_12_bin 48.87784195
ps_ind_13_bin 20.375658989999998
ps_ind_11_bin 9.09727097
ps_ind_14 5.2607857
ps_ind_10_bin 4.19456196
0 1000 2000 3000 4000 5000


TOTAL_COVER
ps_car_13 3917029.06219852
ps_reg_03 1689337.9969175106
ps_ind_05_cat 1000388.06224369
ps_ind_17_bin 674443.2777110501
ps_ind_03 668383.2198490095
ps_reg_02 626693.1180598398
ps_car_14 568578.1110395202
ps_reg_01 519942.11131749017
ps_car_11_cat 517340.38175347
ps_car_07_cat 470682.55913406
ps_ind_15 455667.85583052004
ps_calc_10 405394.60822859
ps_car_01_cat 354887.8303364599
ps_car_12 308529.03584878007
ps_ind_01 284730.54220265994
ps_calc_14 281598.594159
ps_ind_06_bin 264324.51870461006
ps_car_09_cat 256590.69219802006
ps_calc_11 234113.66333567
ps_car_15 215832.53765803
ps_car_06_cat 175374.47554954002
ps_ind_02_cat 157902.78147282

In [32]:
```
%%time
mf = mutual_info_classif(train_imp.drop(['id','target'],axis=1).values,
                         train_imp.target.values,
                         n_neighbors=30, random_state=2022)
print(mf)
```

```
[1.13688315e-03 4.22158074e-03 1.35599884e-03 1.68686989e-03
 8.57707895e-04 1.86228440e-03 1.12560973e-03 2.77409873e-04
 4.06129362e-04 0.00000000e+00 1.14051723e-05 7.66921627e-05
 0.00000000e+00 1.12701163e-05 1.24501323e-03 4.49947066e-03
 5.92141362e-04 2.36087711e-04 1.34802166e-03 9.28019750e-04
 5.18843733e-04 2.80119096e-03 6.64504597e-03 7.93398089e-04
 1.53250383e-03 8.38246069e-03 6.60925418e-03 5.06933218e-03
 9.00388241e-03 1.44670132e-03 3.91695305e-03 1.48370371e-03
 1.84021357e-03 9.77655896e-04 1.26728143e-03 3.82313257e-04
 3.90869701e-04 2.84671759e-04 1.96475285e-03 2.05495176e-03
 1.96014168e-03 1.60294717e-03 1.88351358e-03 1.79755088e-03
 9.60430682e-04 9.10600275e-04 1.74713733e-03 1.38065000e-03
 8.71062322e-04 1.20886246e-04 3.56145372e-03 2.70045648e-03
 7.07345176e-04 1.10377568e-03 9.43819366e-05]
Wall time: 6min 54s
```

In [33]:
```
threshold = 10
high_score_features = []
high_score = []
for score, f_name in sorted(zip(mf, train_imp.drop(['id','target'],axis=1).
    print(f_name, score)
    high_score_features.append(f_name)
    high_score.append(score)
```

```
ps_car_10_cat 0.009003882411455333
ps_car_07_cat 0.008382460694422278
ps_car_02_cat 0.006645045972727637
ps_car_08_cat 0.006609254183864266
ps_car_09_cat 0.005069332176625085
ps_ind_16_bin 0.0044994706649483796
ps_ind_02_cat 0.00422158074290202
ps_car_11 0.003916953051236849
ps_calc_16_bin 0.00356145371805372
ps_car_01_cat 0.0028011909644849453
```

In [34]:
```
sns.barplot(y = high_score_features, x = high_score, palette = 'crest')
```

Out[34]: <AxesSubplot:>