In [8]:
```
# 全集
fullset = pd.concat([train,test],ignore_index=True)
```

In [14]:
```
def meta(train,test,missing_values = -1,cols_ignore_missing = []):

    df = pd.concat([train,test]).reset_index(drop=True).fillna('未知')
    data = []
    for col in df.columns:
        # 定义role
        if col == 'target':
            role = 'target'
        elif col == 'id':
            role = 'id'
        else:
            role = 'feature'

        # 定义category
        if 'ind' in col:
            category = 'individual'
        elif 'car' in col:
            category = 'car'
        elif 'calc' in col:
            category = 'calculated'
        elif 'reg' in col:
            category = 'region'
        else:
            category = 'other'


        # 定义 level of measurements
        if 'bin' in col or col == 'target':
            level = 'binary'
        elif 'cat' in col[-3:] or col == 'id':
            level = 'nominal'
        elif df[col].dtype == 'float64' and df[col].replace(missing_values,
            level = 'interval'
        elif df[col].dtype == 'float64' and df[col].replace(missing_values,
            level = 'ratio'
        elif df[col].dtype == 'int64':
            level = 'ordinal'

        # 定义 data type
        dtype = df[col].dtype

        # 定义 unique
        if col == 'id' or df[col].dtype == 'float64':
            uniq = 'Ignore'
        else:
            if col in cols_ignore_missing:
                uniq = df[col].nunique()
            else:
                uniq = df[col].replace({missing_values:np.nan}).nunique()

        # 定义 cardinality
        if uniq == 'Ignore':
            cardinality = 'Ignore'
        elif uniq <= 10:
            cardinality = 'Low Cardinality'
        elif uniq <= 30:
```

```
            cardinality = 'Medium Cardinality'
        else:
            cardinality = 'High Cardinality'

        # 定义 missing
        if col in cols_ignore_missing:
            missing = 0
        else:
            missing = sum(df[col] == missing_values)

        # 定义 missing percent
        missing_percent = f'{missing}({round(missing*100/len(df),2)}%)'

        # 定义 imputation
        if missing > df.shape[0]*0.4:
            imputation = 'remove'
        elif missing > 0:
            if level == 'binary' or level == 'nominal':
                imputation = ('mode')
            if level == 'ordinal':
                imputation = ('mode','median')
            if level == 'interval' or level == 'ratio':
                imputation = ('mode','median','mean')
        else:
            imputation = "No Missing"

        # 定义 keep
        keep = True
        if col  == 'id' or imputation == 'remove':
            keep = False
        col_dict = {
            'colname': col,
            'role': role,
            'category': category,
            'level': level,
            'dtype': dtype,
            'cardinality': uniq,
            'cardinality_level':cardinality,
            'missing': missing,
            'missing_percent': missing_percent,
            'imputation':imputation,
            'keep': keep,
        }
        data.append(col_dict)
    meta = pd.DataFrame(data, columns=list(col_dict.keys()))
    meta.set_index('colname', inplace=True)

    return meta
```

In [15]:
```
metadata = meta(train,test)
```

In [16]:
```
missing_data = metadata[['missing','missing_percent','imputation']][metadat
```

In [17]:
```
missing_data
```

Out[17]:

colname	missing	missing_percent	imputation
ps_car_03_cat	1028142	1028142(69.09%)	remove
ps_car_05_cat	666910	666910(44.82%)	remove
ps_reg_03	269456	269456(18.11%)	(mode, median, mean)
ps_car_14	106425	106425(7.15%)	(mode, median, mean)
ps_car_07_cat	28820	28820(1.94%)	mode
ps_ind_05_cat	14519	14519(0.98%)	mode
ps_car_09_cat	1446	1446(0.1%)	mode
ps_ind_02_cat	523	523(0.04%)	mode
ps_car_01_cat	267	267(0.02%)	mode
ps_ind_04_cat	228	228(0.02%)	mode
ps_car_02_cat	10	10(0.0%)	mode
ps_car_11	6	6(0.0%)	(mode, median)
ps_car_12	1	1(0.0%)	(mode, median, mean)

In [22]:
```
cols_to_drop = missing_data[missing_data.imputation == 'remove'].index.to_l
```

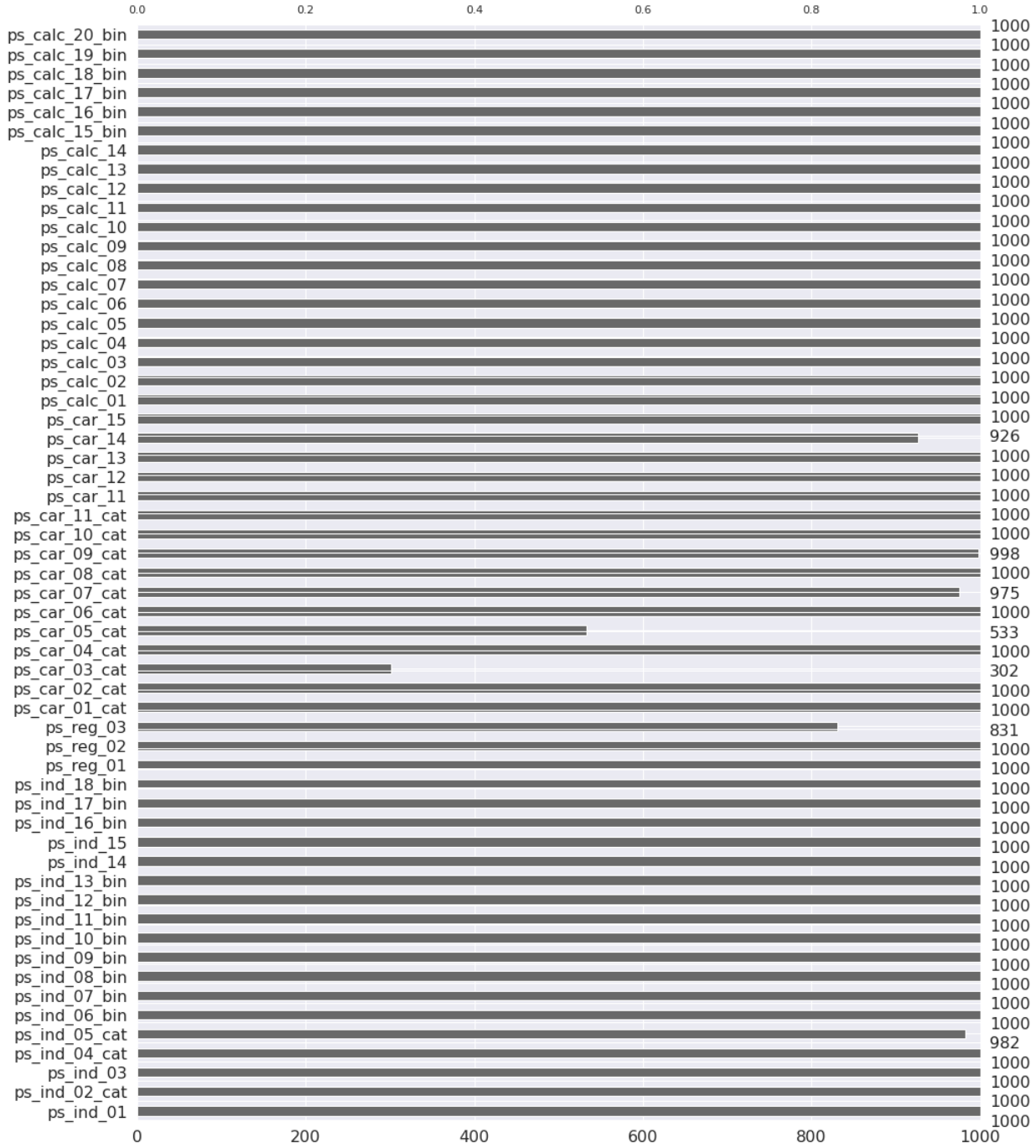
In [23]:
```
cols_to_imp = missing_data.index[2:].to_list()
cols_to_imp_3m = missing_data[missing_data.imputation == ('mode', 'median',
cols_to_imp_2m = missing_data[missing_data.imputation == ('mode', 'median')
cols_to_imp_1m = missing_data[missing_data.imputation == ('mode')].index.to
```

In [24]:
```
import missingno as msno
```

In [25]:
```
msno.bar(fullset.drop(['id','target'],axis=1).replace(-1,np.nan).sample(100
```

Out[25]: <AxesSubplot:>

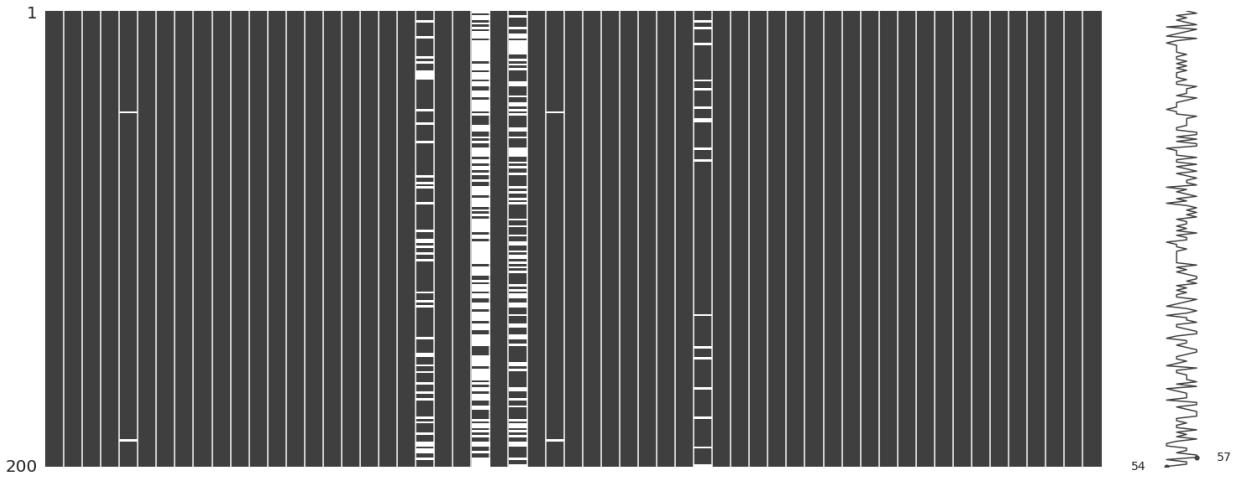
```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sa
ns.
findfont: Generic family 'sans-serif' not found because none of the follo
wing families were found: SimHei
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sa
ns.
findfont: Generic family 'sans-serif' not found because none of the follo
wing families were found: SimHei
```



In [26]:
```
msno.matrix(fullset.drop(['id','target'],axis=1).sample(200).replace(-1,np.
```

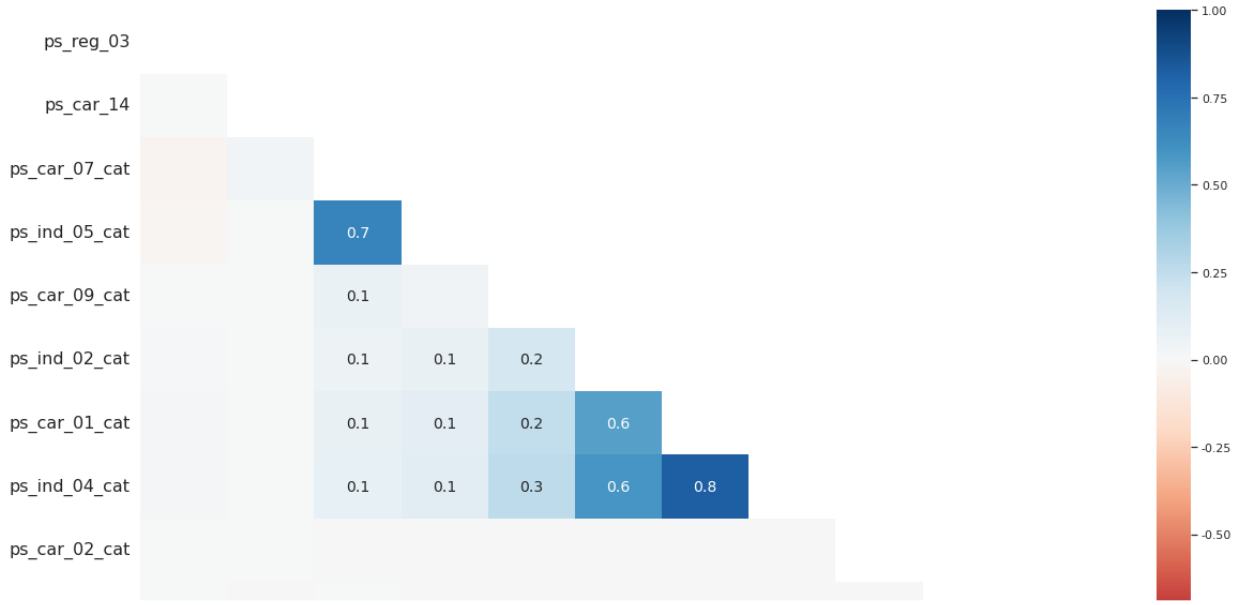
Out[26]: <AxesSubplot:>

```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sa
ns.
findfont: Generic family 'sans-serif' not found because none of the follo
wing families were found: SimHei
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sa
ns.
findfont: Generic family 'sans-serif' not found because none of the follo
wing families were found: SimHei
```



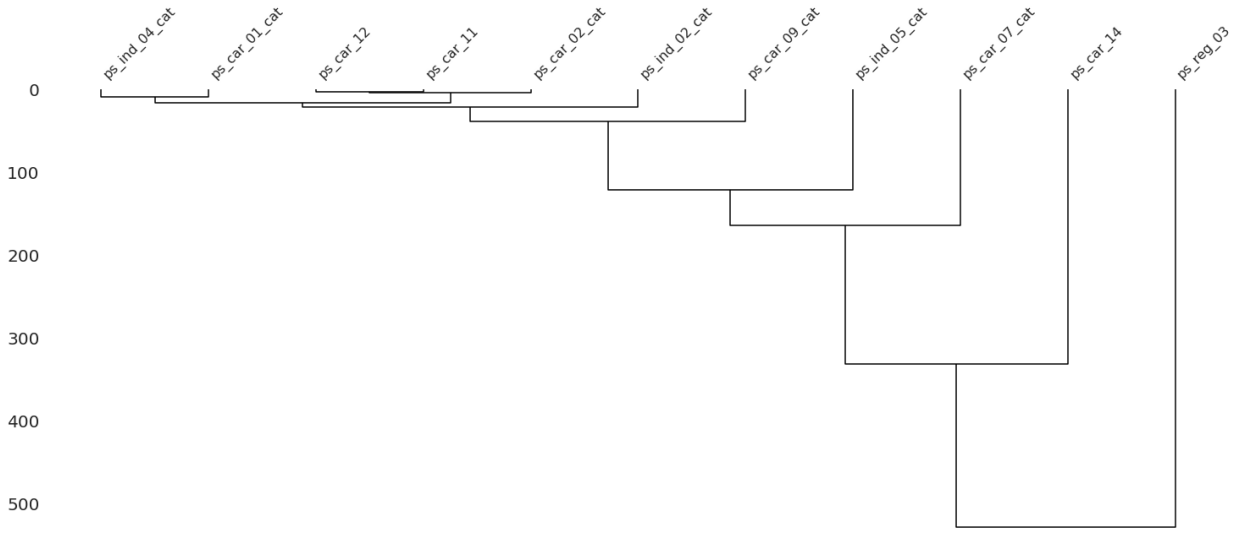
In [28]:
```
msno.heatmap(fullset[cols_to_imp].replace(-1,np.nan))
```

Out[28]: <AxesSubplot:>



In [29]:
```
msno.dendrogram(fullset[cols_to_imp].replace(-1,np.nan))
```

Out[29]: <AxesSubplot:>



In [30]:
```
set1 = ['ps_ind_05_cat','ps_car_07_cat']
set2 = ['ps_car_01_cat','ps_ind_02_cat','ps_ind_04_cat']
set3 = ['ps_reg_03','ps_car_14']
```

In [31]:
```
IFrame(width="853",height="480",src = "https://www.youtube.com/embed/WPiYOS
```

Out[31]:

In [32]:
```
%%time

from sklearn.experimental import enable_iterative_imputer  # noqa
# now you can import normally from sklearn.impute
from sklearn.impute import IterativeImputer

from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=10, random_state=123)

imp_mean = IterativeImputer(estimator=rf, missing_values=-1, random_state=0
set1_imp = imp_mean.fit_transform(train[set1])
```

```
CPU times: user 2.29 s, sys: 211 ms, total: 2.5 s
Wall time: 2.8 s
```

In [33]:
```
%%time
set2_imp = imp_mean.fit_transform(train[set2])
```

```
CPU times: user 10.6 s, sys: 344 ms, total: 10.9 s
Wall time: 11.3 s
```

In [34]:
```
%%time
set3_imp = imp_mean.fit_transform(train[set3])
```

```
CPU times: user 1min 36s, sys: 1.31 s, total: 1min 37s
Wall time: 1min 42s

/Users/mac/opt/anaconda3/lib/python3.9/site-packages/sklearn/impute/_iter
ative.py:713: ConvergenceWarning:

[IterativeImputer] Early stopping criterion not reached.
```

In [35]:
```
pd.DataFrame(set3_imp,columns = set3)
```

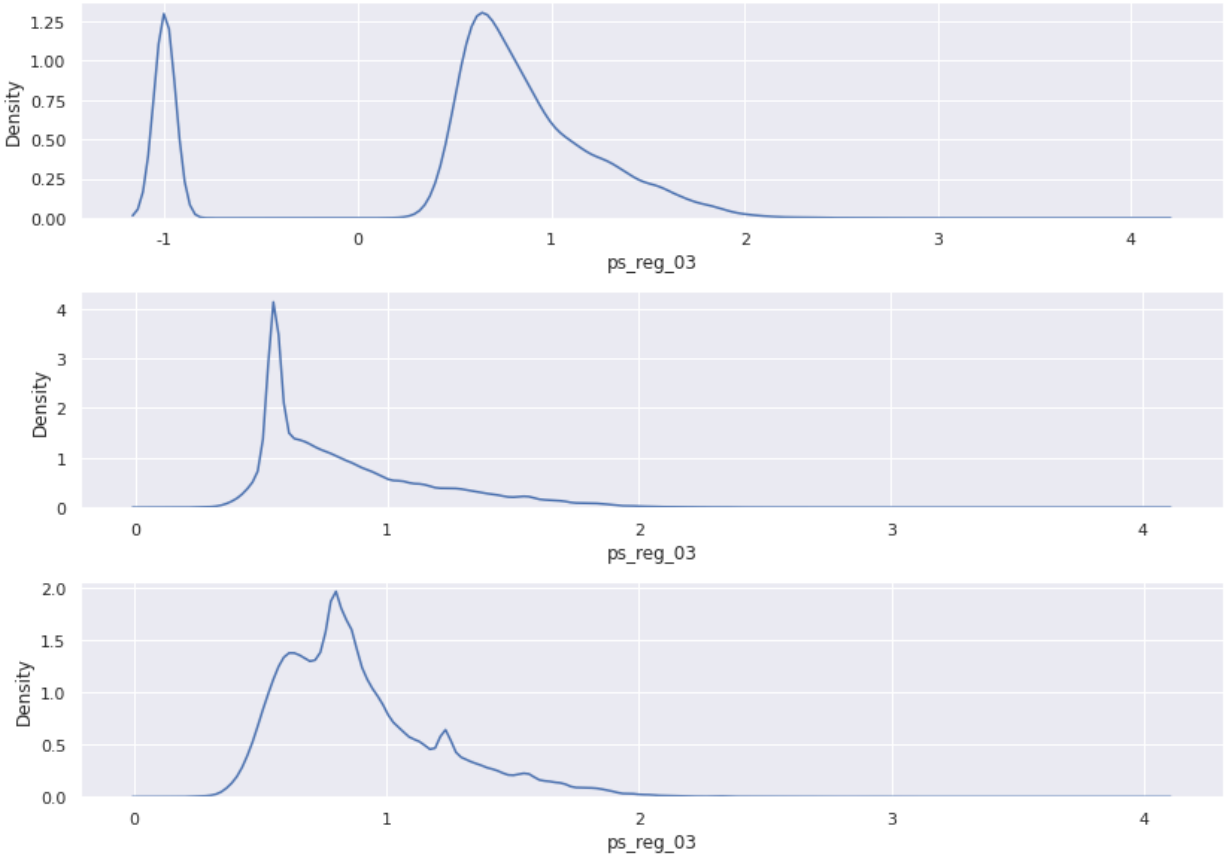
Out[35]:

	ps_reg_03	ps_car_14
0	0.718070	0.370810
1	0.766078	0.388716
2	0.855884	0.347275
3	0.580948	0.294958
4	0.840759	0.365103
...	...	...
595207	0.692820	0.385487
595208	1.382027	0.378471
595209	0.659071	0.398748
595210	0.698212	0.384968
595211	0.776784	0.378021

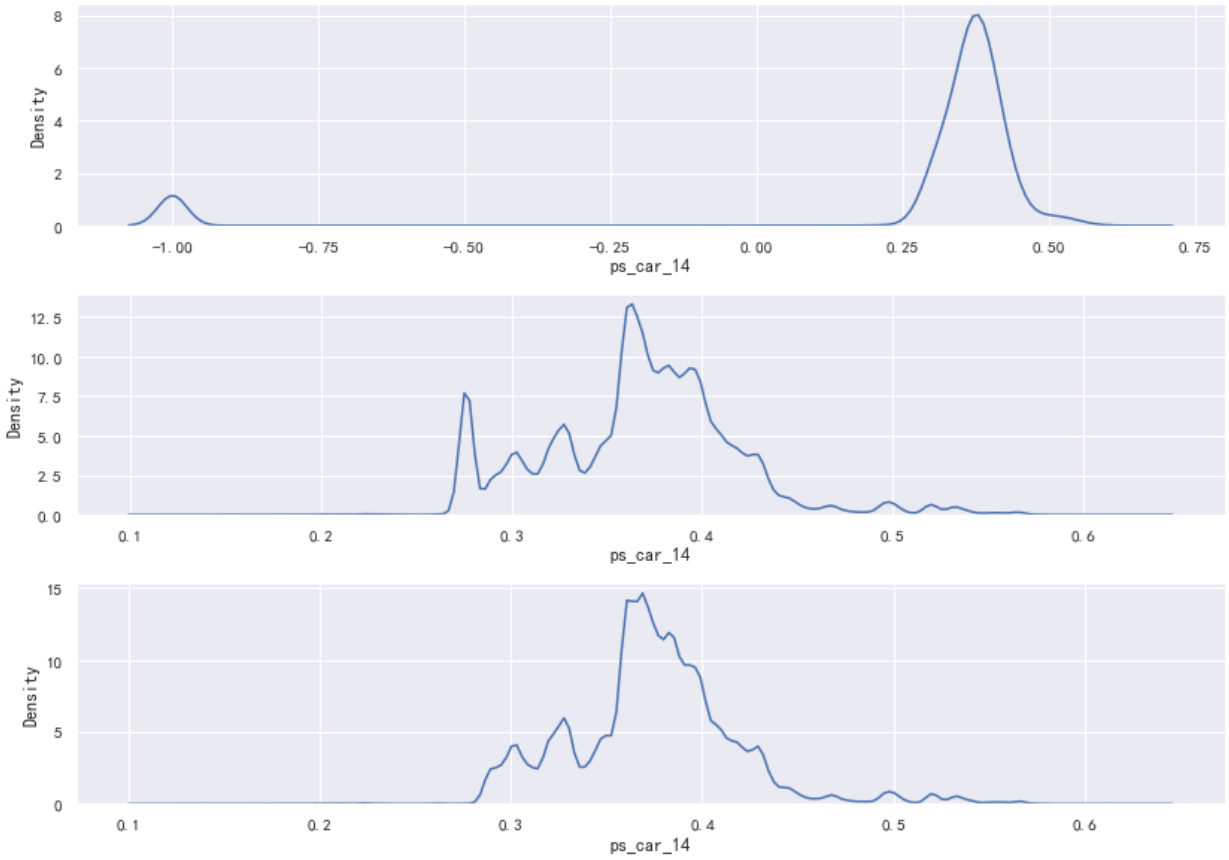
595212 rows × 2 columns

In [36]:
```
plt.subplot(3,1,1)
sns.kdeplot(train[set3[0]])
plt.subplot(3,1,2)
sns.kdeplot(train[set3[0]].replace(-1,np.nan).fillna(train[set3[0]].mean()))
plt.subplot(3,1,3)
sns.kdeplot(pd.DataFrame(set3_imp,columns = set3)[set3[0]])

plt.tight_layout()
```

```
findfont: Font family ['sans-serif'] not found. Falling back to DejaVu Sa
ns.
findfont: Generic family 'sans-serif' not found because none of the follo
wing families were found: SimHei
```



In [22]:
```
plt.subplot(3,1,1)
sns.kdeplot(train[set3[1]])
plt.subplot(3,1,2)
sns.kdeplot(train[set3[1]].replace(-1,np.nan).fillna(train[set3[1]].mean()))
plt.subplot(3,1,3)
sns.kdeplot(pd.DataFrame(set3_imp,columns = set3)[set3[1]])

plt.tight_layout()
```



In [37]:
```
train[set3] = pd.DataFrame(set3_imp,columns = set3)
```

KNN

In [38]:
```
%%time
# from sklearn.impute import KNNImputer
# imputer = KNNImputer(missing_values=-1,n_neighbors=2000)
# imputer.fit_transform(train[set1])

# Wall time: 7min 35s
```

```
CPU times: user 2 µs, sys: 1 µs, total: 3 µs
Wall time: 5.01 µs
```

In [42]:
```
from sklearn.impute import SimpleImputer
```

In [43]:
```
mode_imputer = SimpleImputer(missing_values = -1, strategy='most_frequent',
```

In [51]:
```
imp = mode_imputer.fit_transform(train[missing_data[4:].index])
```

In [55]:
```
train[missing_data[4:].index]=pd.DataFrame(imp).iloc[:,:len(missing_data[4:
```

In [57]:
```
train[missing_data[4:].index].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 595212 entries, 0 to 595211
Data columns (total 9 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   ps_car_07_cat   595212 non-null  float64
 1   ps_ind_05_cat   595212 non-null  float64
 2   ps_car_09_cat   595212 non-null  float64
 3   ps_ind_02_cat   595212 non-null  float64
 4   ps_car_01_cat   595212 non-null  float64
 5   ps_ind_04_cat   595212 non-null  float64
 6   ps_car_02_cat   595212 non-null  float64
 7   ps_car_11       595212 non-null  float64
 8   ps_car_12       595212 non-null  float64
dtypes: float64(9)
memory usage: 40.9 MB
```

In [58]:
```
train[missing_data[4:].index] = train[missing_data[4:].index].astype('int64
```

In [34]:
```
# drop columns
train.drop(cols_to_drop,axis=1,inplace=True)
```

In [35]:
```
# check out if we still have -1
(train == -1).sum().sum()
```

Out[35]: 0

In [36]:
```
train.to_csv('train_imp.csv')
```