m04_v01_store_sales_prediction

September 12, 2021

1 0.0. IMPORTS

[1]:
```
import math
import numpy  as np
import pandas as pd
import inflection
import seaborn as sns


from scipy                  import stats  as ss
from matplotlib             import pyplot as plt
from IPython.display        import Image
from IPython.core.display   import HTML

from sklearn.preprocessing import RobustScaler, MinMaxScaler, LabelEncoder
```

1.1 0.1. Helper Functions

[2]:
```
def cramer_v( x, y ):
    cm = pd.crosstab( x, y ).as_matrix()
    n = cm.sum()
    r, k = cm.shape

    chi2 = ss.chi2_contingency( cm )[0]
    chi2corr = max( 0, chi2 - (k-1)*(r-1)/(n-1) )

    kcorr = k - (k-1)**2/(n-1)
    rcorr = r - (r-1)**2/(n-1)

    return np.sqrt( (chi2corr/n) / ( min( kcorr-1, rcorr-1 ) ) )



def jupyter_settings():
    %matplotlib inline
    %pylab inline
```

```
    plt.style.use( 'bmh' )
    plt.rcParams['figure.figsize'] = [25, 12]
    plt.rcParams['font.size'] = 24

    display( HTML( '<style>.container { width:100% !important; }</style>') )
    pd.options.display.max_columns = None
    pd.options.display.max_rows = None
    pd.set_option( 'display.expand_frame_repr', False )

    sns.set()
```

[3]:
```
jupyter_settings()
```

```
Populating the interactive namespace from numpy and matplotlib
<IPython.core.display.HTML object>
```

1.2 0.2. Loading data

[5]:
```
df_sales_raw = pd.read_csv( '../data/train.csv', low_memory=False )
df_store_raw = pd.read_csv( '../data/store.csv', low_memory=False )

# merge
df_raw = pd.merge( df_sales_raw, df_store_raw, how='left', on='Store' )
```

2 1.0. PASSO 01 - DESCRICAO DOS DADOS

[5]:
```
df1 = df_raw.copy()
```

2.1 1.1. Rename Columns

[6]:
```
cols_old = ['Store', 'DayOfWeek', 'Date', 'Sales', 'Customers', 'Open',␣
 ↪'Promo', 'StateHoliday', 'SchoolHoliday',
            'StoreType', 'Assortment', 'CompetitionDistance',␣
 ↪'CompetitionOpenSinceMonth',
            'CompetitionOpenSinceYear', 'Promo2', 'Promo2SinceWeek',␣
 ↪'Promo2SinceYear', 'PromoInterval']

snakecase = lambda x: inflection.underscore( x )

cols_new = list( map( snakecase, cols_old ) )

# rename
df1.columns = cols_new
```

2.2 1.2. Data Dimensions

[7]:
```
print( 'Number of Rows: {}'.format( df1.shape[0] ) )
print( 'Number of Cols: {}'.format( df1.shape[1] ) )
```

```
Number of Rows: 1017209
Number of Cols: 18
```

2.3 1.3. Data Types

[8]:
```
df1['date'] = pd.to_datetime( df1['date'] )
df1.dtypes
```

[8]:
```
store                                    int64
day_of_week                              int64
date                            datetime64[ns]
sales                                    int64
customers                                int64
open                                     int64
promo                                    int64
state_holiday                           object
school_holiday                           int64
store_type                              object
assortment                              object
competition_distance                   float64
competition_open_since_month           float64
competition_open_since_year            float64
promo2                                   int64
promo2_since_week                      float64
promo2_since_year                      float64
promo_interval                          object
dtype: object
```

2.4 1.4. Check NA

[9]:
```
df1.isna().sum()
```

[9]:
```
store                           0
day_of_week                     0
date                            0
sales                           0
customers                       0
open                            0
promo                           0
state_holiday                   0
school_holiday                  0
store_type                      0
assortment                      0
```

```
competition_distance             2642
competition_open_since_month   323348
competition_open_since_year    323348
promo2                              0
promo2_since_week              508031
promo2_since_year              508031
promo_interval                 508031
dtype: int64
```

2.5 1.5. Fillout NA

[10]:
```python
df1.sample()
```

[10]:
```
        store  day_of_week       date  sales  customers  open  promo
state_holiday  school_holiday store_type assortment  competition_distance
competition_open_since_month  competition_open_since_year  promo2
promo2_since_week  promo2_since_year   promo_interval
519202    398            7 2014-03-23      0          0     0      0
0               0          c          c                1540.0
NaN                            NaN       1                 1.0             2012.0
Jan,Apr,Jul,Oct
```

[11]:
```python
#competition_distance
df1['competition_distance'] = df1['competition_distance'].apply( lambda x:␣
 ↪200000.0 if math.isnan( x ) else x )

#competition_open_since_month
df1['competition_open_since_month'] = df1.apply( lambda x: x['date'].month if␣
 ↪math.isnan( x['competition_open_since_month'] ) else␣
 ↪x['competition_open_since_month'], axis=1 )

#competition_open_since_year
df1['competition_open_since_year'] = df1.apply( lambda x: x['date'].year if␣
 ↪math.isnan( x['competition_open_since_year'] ) else␣
 ↪x['competition_open_since_year'], axis=1 )

#promo2_since_week
df1['promo2_since_week'] = df1.apply( lambda x: x['date'].week if math.isnan(␣
 ↪x['promo2_since_week'] ) else x['promo2_since_week'], axis=1 )

#promo2_since_year
df1['promo2_since_year'] = df1.apply( lambda x: x['date'].year if math.isnan(␣
 ↪x['promo2_since_year'] ) else x['promo2_since_year'], axis=1 )

#promo_interval
month_map = {1: 'Jan',  2: 'Fev',  3: 'Mar',  4: 'Apr',  5: 'May',  6: 'Jun', ␣
 ↪7: 'Jul',  8: 'Aug',  9: 'Sep',  10: 'Oct', 11: 'Nov', 12: 'Dec'}
```

```python
df1['promo_interval'].fillna(0, inplace=True )

df1['month_map'] = df1['date'].dt.month.map( month_map )

df1['is_promo'] = df1[['promo_interval', 'month_map']].apply( lambda x: 0 if␣
↪x['promo_interval'] == 0 else 1 if x['month_map'] in x['promo_interval'].
↪split( ',' ) else 0, axis=1 )
```

[12]:
```python
df1.isna().sum()
```

[12]:
```
store                           0
day_of_week                     0
date                            0
sales                           0
customers                       0
open                            0
promo                           0
state_holiday                   0
school_holiday                  0
store_type                      0
assortment                      0
competition_distance            0
competition_open_since_month    0
competition_open_since_year     0
promo2                          0
promo2_since_week               0
promo2_since_year               0
promo_interval                  0
month_map                       0
is_promo                        0
dtype: int64
```

2.6 1.6. Change Data Types

[13]:
```python
# competiton
df1['competition_open_since_month'] = df1['competition_open_since_month'].
↪astype( int )
df1['competition_open_since_year'] = df1['competition_open_since_year'].astype(␣
↪int )

# promo2
df1['promo2_since_week'] = df1['promo2_since_week'].astype( int )
df1['promo2_since_year'] = df1['promo2_since_year'].astype( int )
```

2.7 1.7. Descriptive Statistics

[14]:
```
num_attributes = df1.select_dtypes( include=['int64', 'float64'] )
cat_attributes = df1.select_dtypes( exclude=['int64', 'float64',␣
↪'datetime64[ns]'] )
```

2.7.1 1.7.1. Numerical Atributes

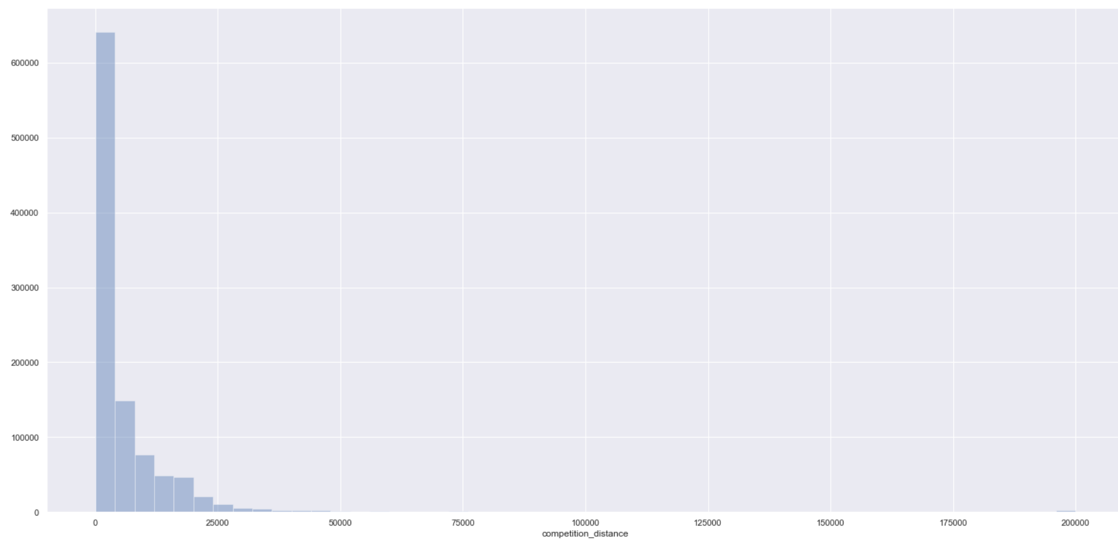
[15]:
```
# Central Tendency - mean, meadina
ct1 = pd.DataFrame( num_attributes.apply( np.mean ) ).T
ct2 = pd.DataFrame( num_attributes.apply( np.median ) ).T

# dispersion - std, min, max, range, skew, kurtosis
d1 = pd.DataFrame( num_attributes.apply( np.std ) ).T
d2 = pd.DataFrame( num_attributes.apply( min ) ).T
d3 = pd.DataFrame( num_attributes.apply( max ) ).T
d4 = pd.DataFrame( num_attributes.apply( lambda x: x.max() - x.min() ) ).T
d5 = pd.DataFrame( num_attributes.apply( lambda x: x.skew() ) ).T
d6 = pd.DataFrame( num_attributes.apply( lambda x: x.kurtosis() ) ).T

# concatenar
m = pd.concat( [d2, d3, d4, ct1, ct2, d1, d5, d6] ).T.reset_index()
m.columns = ['attributes', 'min', 'max', 'range', 'mean', 'median', 'std',␣
↪'skew', 'kurtosis']
m
```

[15]:
```
                      attributes     min       max     range          mean  \
median           std        skew    kurtosis
0                          store     1.0    1115.0    1114.0    558.429727
558.0    321.908493   -0.000955   -1.200524
1                    day_of_week     1.0       7.0       6.0      3.998341
4.0      1.997390    0.001593   -1.246873
2                          sales     0.0   41551.0   41551.0   5773.818972
5744.0   3849.924283    0.641460    1.778375
3                      customers     0.0    7388.0    7388.0    633.145946
609.0    464.411506    1.598650    7.091773
4                           open     0.0       1.0       1.0      0.830107
1.0      0.375539   -1.758045    1.090723
5                          promo     0.0       1.0       1.0      0.381515
0.0      0.485758    0.487838   -1.762018
6                 school_holiday     0.0       1.0       1.0      0.178647
0.0      0.383056    1.677842    0.815154
7           competition_distance    20.0  200000.0  199980.0   5935.442677
2330.0  12547.646829   10.242344  147.789712
8   competition_open_since_month     1.0      12.0      11.0      6.786849
7.0      3.311085   -0.042076   -1.232607
9    competition_open_since_year  1900.0    2015.0     115.0   2010.324840
```

```
2012.0      5.515591  -7.235657  124.071304
10                         promo2     0.0       1.0       1.0     0.500564
1.0      0.500000  -0.002255   -1.999999
11              promo2_since_week     1.0      52.0      51.0    23.619033
22.0     14.310057   0.178723   -1.184046
12              promo2_since_year  2009.0    2015.0       6.0  2012.793297
2013.0      1.662657  -0.784436   -0.210075
13                       is_promo     0.0       1.0       1.0     0.155231
0.0      0.362124   1.904152    1.625796
```

[16]:
```python
sns.distplot( df1['competition_distance'], kde=False )
```

[16]: <matplotlib.axes._subplots.AxesSubplot at 0x157ae2520>



2.7.2 1.7.2. Categorical Atributes

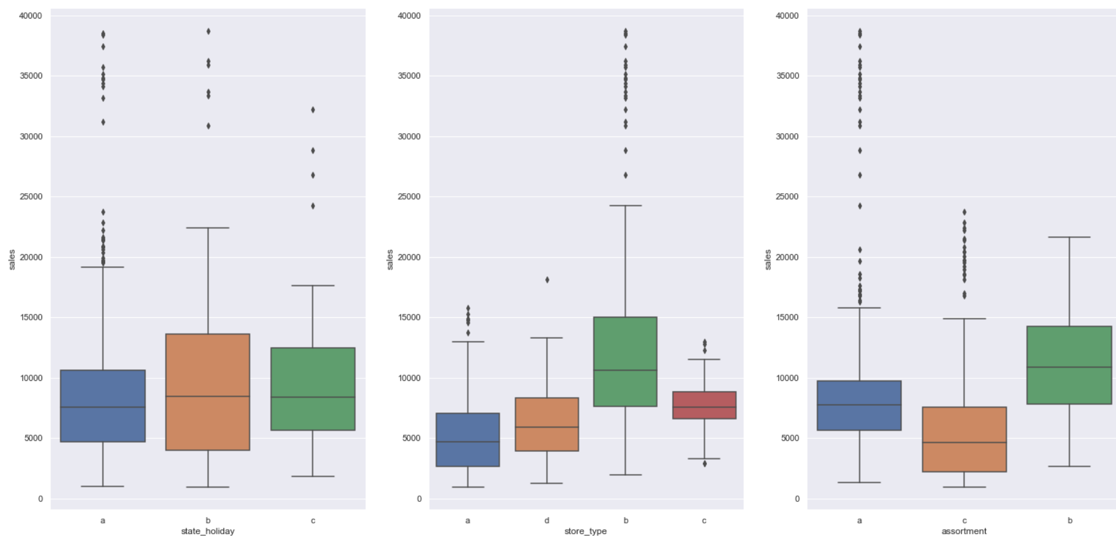
[17]:
```python
cat_attributes.apply( lambda x: x.unique().shape[0] )
```

[17]:
```
state_holiday      4
store_type         4
assortment         3
promo_interval     4
month_map         12
dtype: int64
```

[18]:
```python
aux = df1[(df1['state_holiday'] != '0') & (df1['sales'] > 0)]

plt.subplot( 1, 3, 1 )
sns.boxplot( x='state_holiday', y='sales', data=aux )
```

```python
plt.subplot( 1, 3, 2 )
sns.boxplot( x='store_type', y='sales', data=aux )

plt.subplot( 1, 3, 3 )
sns.boxplot( x='assortment', y='sales', data=aux )
```

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x10312af10>



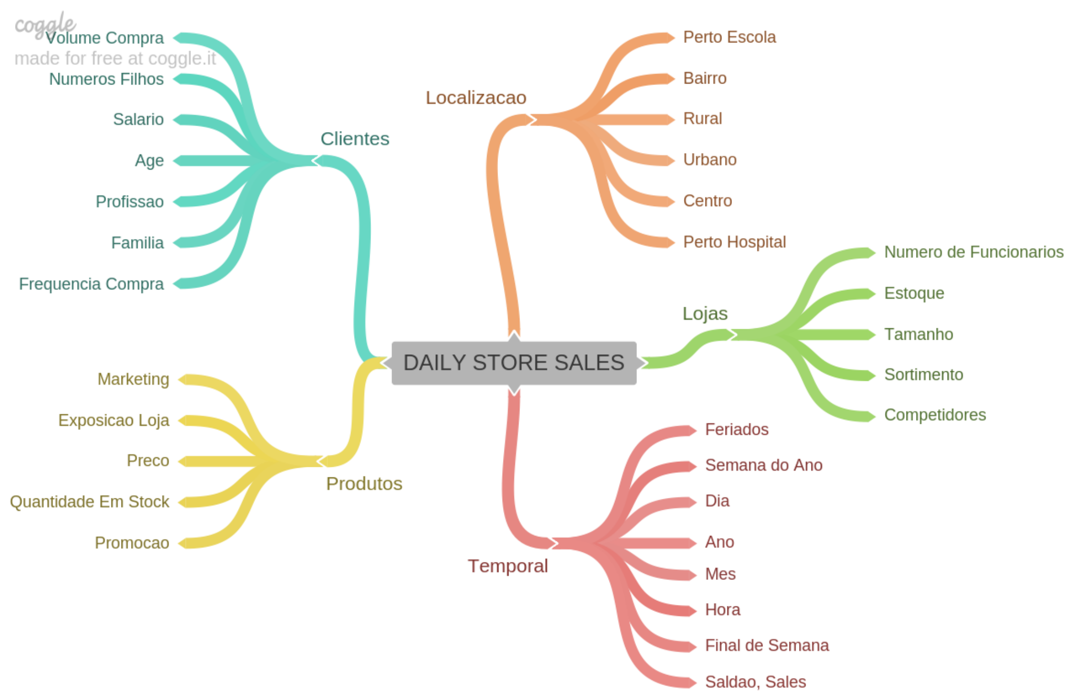
3 2.0. PASSO 02 - FEATURE ENGINEERING

[19]:
```python
df2 = df1.copy()
```

3.1 2.1. Mapa Mental de Hipoteses

[20]:
```python
Image( 'img/MindMapHypothesis.png' )
```

[20]:



3.2 2.2. Criacao das Hipoteses

3.2.1 2.2.1. Hipoteses Loja

1. Lojas com número maior de funcionários deveriam vender mais.

2. Lojas com maior capacidade de estoque deveriam vender mais.

3. Lojas com maior porte deveriam vender mais.

4. Lojas com maior sortimentos deveriam vender mais.

5. Lojas com competidores mais próximos deveriam vender menos.

6. Lojas com competidores à mais tempo deveriam vendem mais.

3.2.2 2.2.2. Hipoteses Produto

1. Lojas que investem mais em Marketing deveriam vender mais.

2. Lojas com maior exposição de produto deveriam vender mais.

3. Lojas com produtos com preço menor deveriam vender mais.

5. Lojas com promoções mais agressivas (descontos maiores), deveriam vender mais.

6. Lojas com promoções ativas por mais tempo deveriam vender mais.

7. Lojas com mais dias de promoção deveriam vender mais.

8. Lojas com mais promoções consecutivas deveriam vender mais.

3.2.3 2.2.3. Hipoteses Tempo

1. Lojas abertas durante o feriado de Natal deveriam vender mais.

2. Lojas deveriam vender mais ao longo dos anos.

3. Lojas deveriam vender mais no segundo semestre do ano.

4. Lojas deveriam vender mais depois do dia 10 de cada mês.

5. Lojas deveriam vender menos aos finais de semana.

6. Lojas deveriam vender menos durante os feriados escolares.

3.3 2.3. Lista Final de Hipóteses

1. Lojas com maior sortimentos deveriam vender mais.

2. Lojas com competidores mais próximos deveriam vender menos.

3. Lojas com competidores à mais tempo deveriam vendem mais.

4. Lojas com promoções ativas por mais tempo deveriam vender mais.

5. Lojas com mais dias de promoção deveriam vender mais.

7. Lojas com mais promoções consecutivas deveriam vender mais.

8. Lojas abertas durante o feriado de Natal deveriam vender mais.

9. Lojas deveriam vender mais ao longo dos anos.

10. Lojas deveriam vender mais no segundo semestre do ano.

11. Lojas deveriam vender mais depois do dia 10 de cada mês.

12. Lojas deveriam vender menos aos finais de semana.

13. Lojas deveriam vender menos durante os feriados escolares.

3.4 2.4. Feature Engineering

[21]:
```
# year
df2['year'] = df2['date'].dt.year

# month
df2['month'] = df2['date'].dt.month

# day
df2['day'] = df2['date'].dt.day

# week of year
df2['week_of_year'] = df2['date'].dt.weekofyear
```

```
# year week
df2['year_week'] = df2['date'].dt.strftime( '%Y-%W' )

# competition since
df2['competition_since'] = df2.apply( lambda x: datetime.datetime(␣
↪year=x['competition_open_since_year'],␣
↪month=x['competition_open_since_month'],day=1 ), axis=1 )
df2['competition_time_month'] = ( ( df2['date'] - df2['competition_since'] )/30␣
↪).apply( lambda x: x.days ).astype( int )

# promo since
df2['promo_since'] = df2['promo2_since_year'].astype( str ) + '-' +␣
↪df2['promo2_since_week'].astype( str )
df2['promo_since'] = df2['promo_since'].apply( lambda x: datetime.datetime.
↪strptime( x + '-1', '%Y-%W-%w' ) - datetime.timedelta( days=7 ) )
df2['promo_time_week'] = ( ( df2['date'] - df2['promo_since'] )/7 ).apply(␣
↪lambda x: x.days ).astype( int )

# assortment
df2['assortment'] = df2['assortment'].apply( lambda x: 'basic' if x == 'a' else␣
↪'extra' if x == 'b' else 'extended' )

# state holiday
df2['state_holiday'] = df2['state_holiday'].apply( lambda x: 'public_holiday'␣
↪if x == 'a' else 'easter_holiday' if x == 'b' else 'christmas' if x == 'c'␣
↪else 'regular_day' )
```

4 3.0. PASSO 03 - FILTRAGEM DE VARIÁVEIS

[22]:
```
df3 = df2.copy()
```

4.1 3.1. Filtragem das Linhas

[23]:
```
df3 = df3[(df3['open'] != 0) & (df3['sales'] > 0)]
```

4.2 3.2. Selecao das Colunas

[24]:
```
cols_drop = ['customers', 'open', 'promo_interval', 'month_map']
df3 = df3.drop( cols_drop, axis=1 )
```

5 4.0. PASSO 04 - ANALISE EXPLORATORIA DOS DADOS

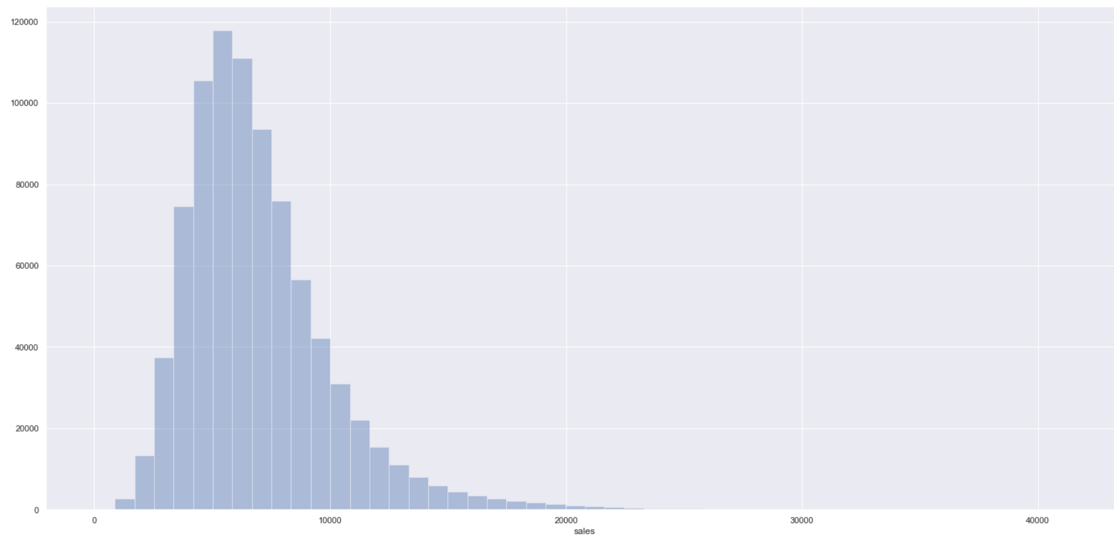
[25]:
```
df4 = df3.copy()
```

5.1 4.1. Analise Univariada

5.1.1 4.1.1. Response Variable

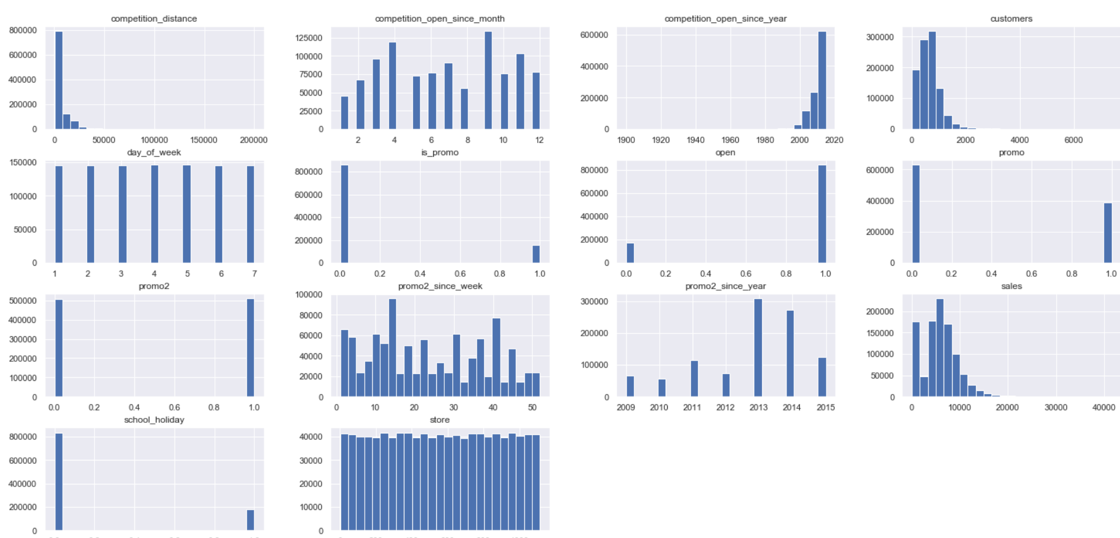
[26]:
```
sns.distplot( df4['sales'], kde=False  )
```

[26]: <matplotlib.axes._subplots.AxesSubplot at 0x179ba10d0>



5.1.2 4.1.2. Numerical Variable

[27]:
```
num_attributes.hist( bins=25 );
```

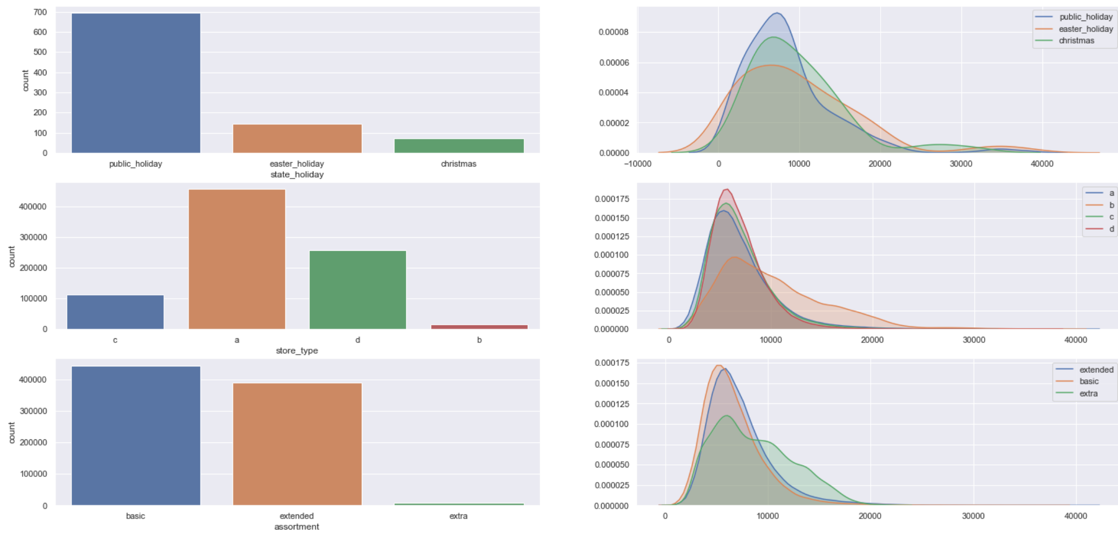


Gabriel Rodrigues Alves - galves.rod@outlook.com - CPF: 086.143.966-00

5.1.3 4.1.3. Categorical Variable

[28]:
```
# state_holiday
plt.subplot( 3, 2, 1 )
a = df4[df4['state_holiday'] != 'regular_day']
sns.countplot( a['state_holiday'] )

plt.subplot( 3, 2, 2 )
sns.kdeplot( df4[df4['state_holiday'] == 'public_holiday']['sales'],␣
 ↪label='public_holiday', shade=True )
sns.kdeplot( df4[df4['state_holiday'] == 'easter_holiday']['sales'],␣
 ↪label='easter_holiday', shade=True )
sns.kdeplot( df4[df4['state_holiday'] == 'christmas']['sales'],␣
 ↪label='christmas', shade=True )

# store_type
plt.subplot( 3, 2, 3 )
sns.countplot( df4['store_type'] )

plt.subplot( 3, 2, 4 )
sns.kdeplot( df4[df4['store_type'] == 'a']['sales'], label='a', shade=True )
sns.kdeplot( df4[df4['store_type'] == 'b']['sales'], label='b', shade=True )
sns.kdeplot( df4[df4['store_type'] == 'c']['sales'], label='c', shade=True )
sns.kdeplot( df4[df4['store_type'] == 'd']['sales'], label='d', shade=True )

# assortment
plt.subplot( 3, 2, 5 )
sns.countplot( df4['assortment'] )

plt.subplot( 3, 2, 6 )
sns.kdeplot( df4[df4['assortment'] == 'extended']['sales'], label='extended',␣
 ↪shade=True )
sns.kdeplot( df4[df4['assortment'] == 'basic']['sales'], label='basic',␣
 ↪shade=True )
sns.kdeplot( df4[df4['assortment'] == 'extra']['sales'], label='extra',␣
 ↪shade=True )
```

[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x167f7f4f0>
```



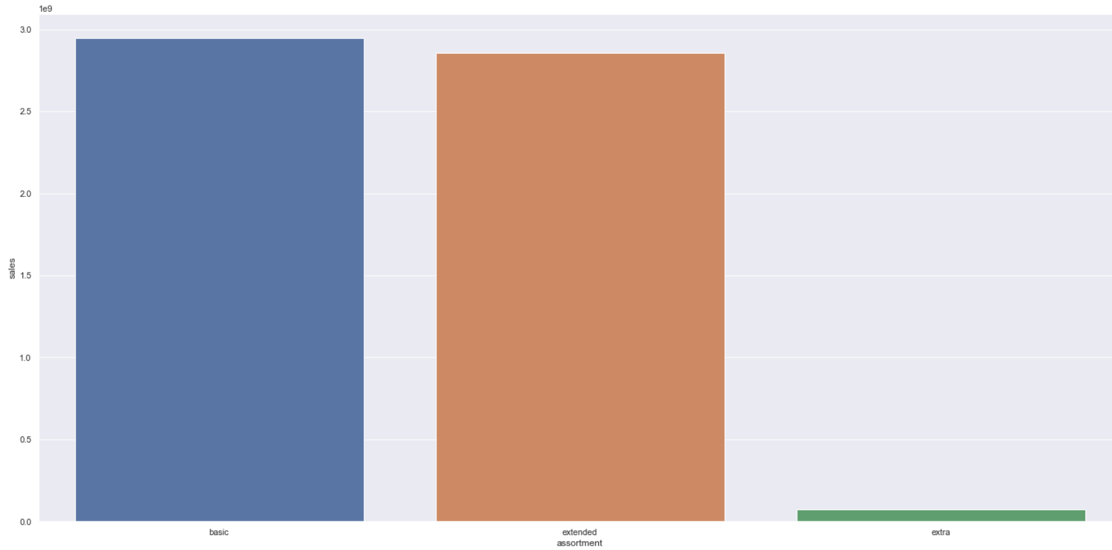
5.2 4.2. Analise Bivariada

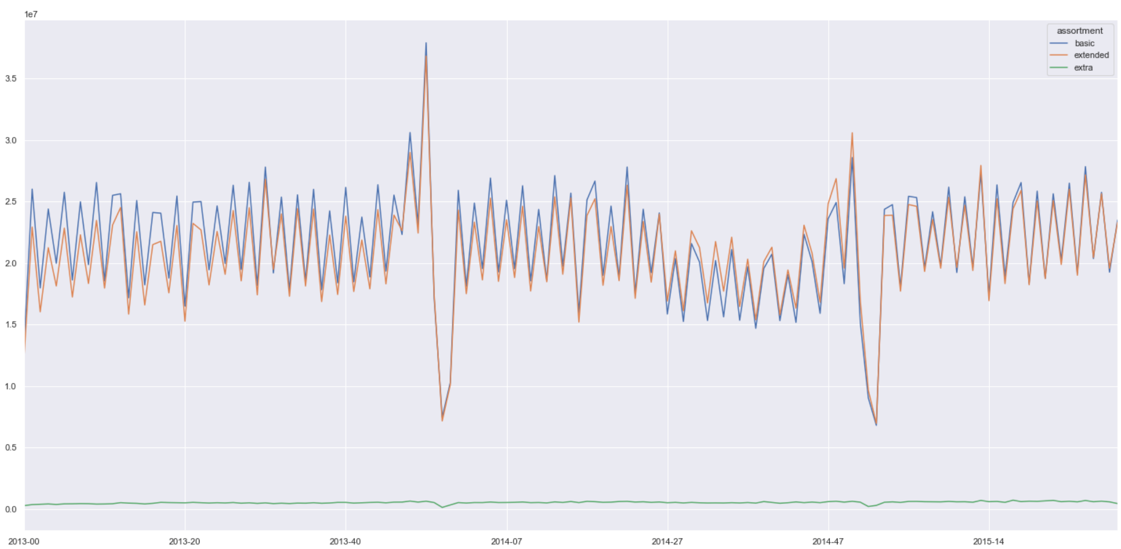
5.2.1 H1. Lojas com maior sortimentos deveriam vender mais.

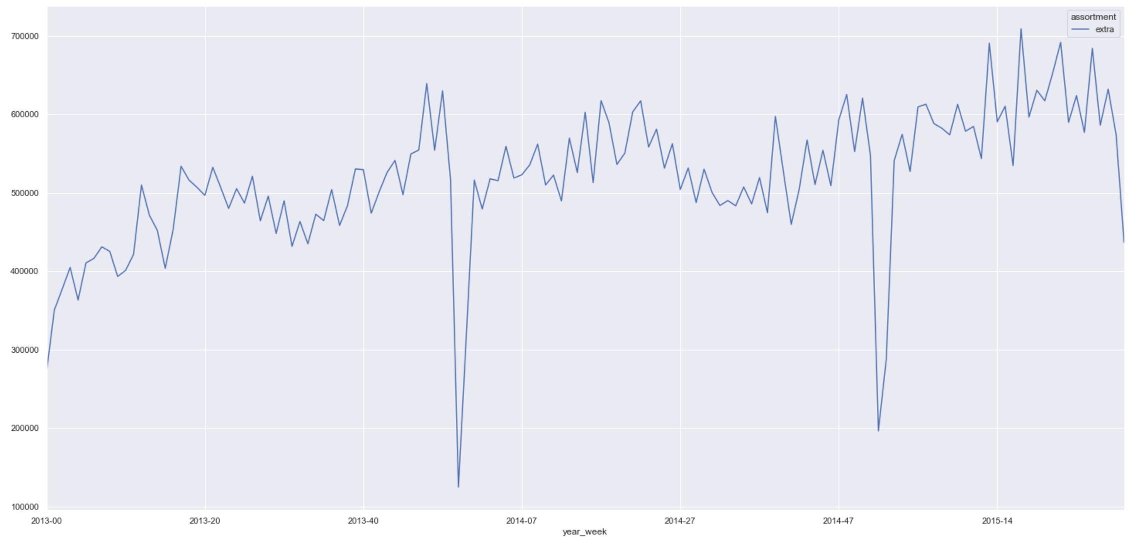
FALSA Lojas com MAIOR SORTIMENTO vendem MENOS.

[29]:
```
aux1 = df4[['assortment', 'sales']].groupby( 'assortment' ).sum().reset_index()
sns.barplot( x='assortment', y='sales', data=aux1 );

aux2 = df4[['year_week', 'assortment', 'sales']].groupby( ['year_week','assortment'] ).sum().reset_index()
aux2.pivot( index='year_week', columns='assortment', values='sales' ).plot()

aux3 = aux2[aux2['assortment'] == 'extra']
aux3.pivot( index='year_week', columns='assortment', values='sales' ).plot()
```

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x11764edc0>


1e9
3.0
2.5
2.0
1.5
1.0
0.5
0.0
sales
basic
extended
extra
assortment

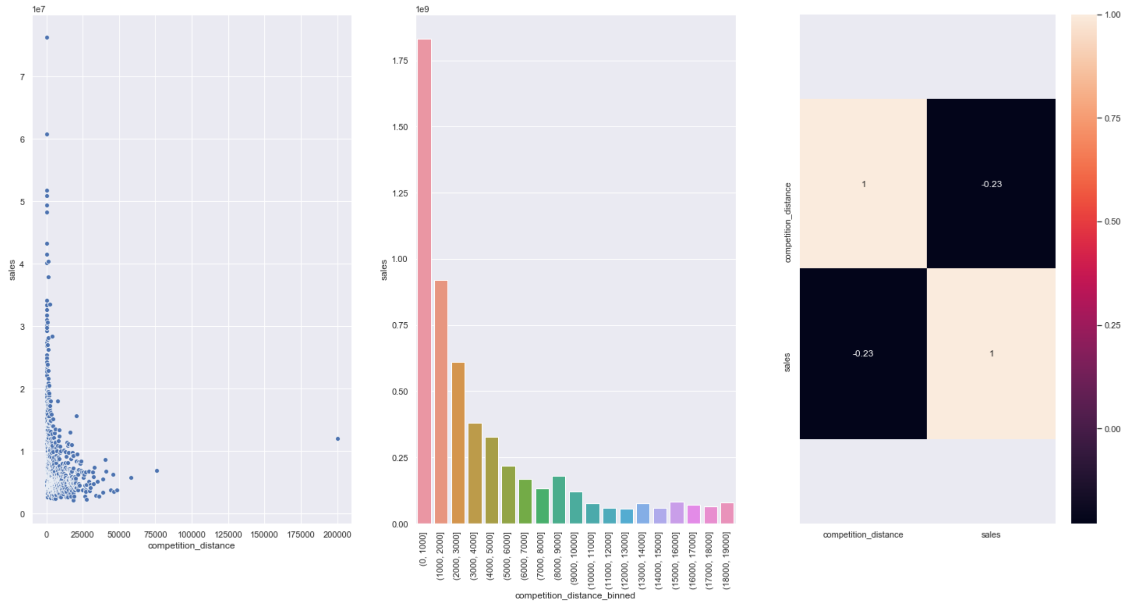

1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
assortment
basic
extended
extra
2013-00
2013-20
2013-40
2014-07
2014-27
2014-47
2015-14
year_week



5.2.2 H2. Lojas com competidores mais próximos deveriam vender menos.

FALSA Lojas com COMPETIDORES MAIS PROXIMOS vendem MAIS.

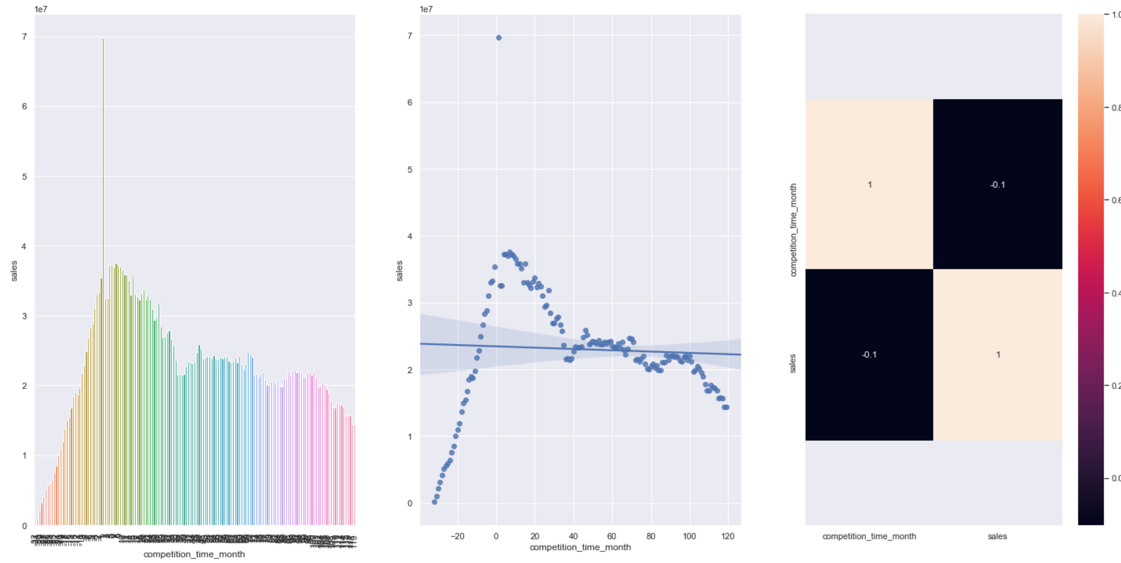
[30]:
```
aux1 = df4[['competition_distance', 'sales']].groupby( 'competition_distance' ).
 ↪sum().reset_index()

plt.subplot( 1, 3, 1 )
sns.scatterplot( x ='competition_distance', y='sales', data=aux1 );

plt.subplot( 1, 3, 2 )
bins = list( np.arange( 0, 20000, 1000) )
aux1['competition_distance_binned'] = pd.cut( aux1['competition_distance'],␣
 ↪bins=bins )
aux2 = aux1[['competition_distance_binned', 'sales']].groupby(␣
 ↪'competition_distance_binned' ).sum().reset_index()
sns.barplot( x='competition_distance_binned', y='sales', data=aux2 );
plt.xticks( rotation=90 );

plt.subplot( 1, 3, 3 )
x = sns.heatmap( aux1.corr( method='pearson' ), annot=True );
bottom, top = x.get_ylim()
x.set_ylim( bottom+0.5, top-0.5 );
```



5.2.3 H3. Lojas com competidores à mais tempo deveriam vendem mais.

FALSE Lojas com COMPETIDORES À MAIS TEMPO vendem MENOS.

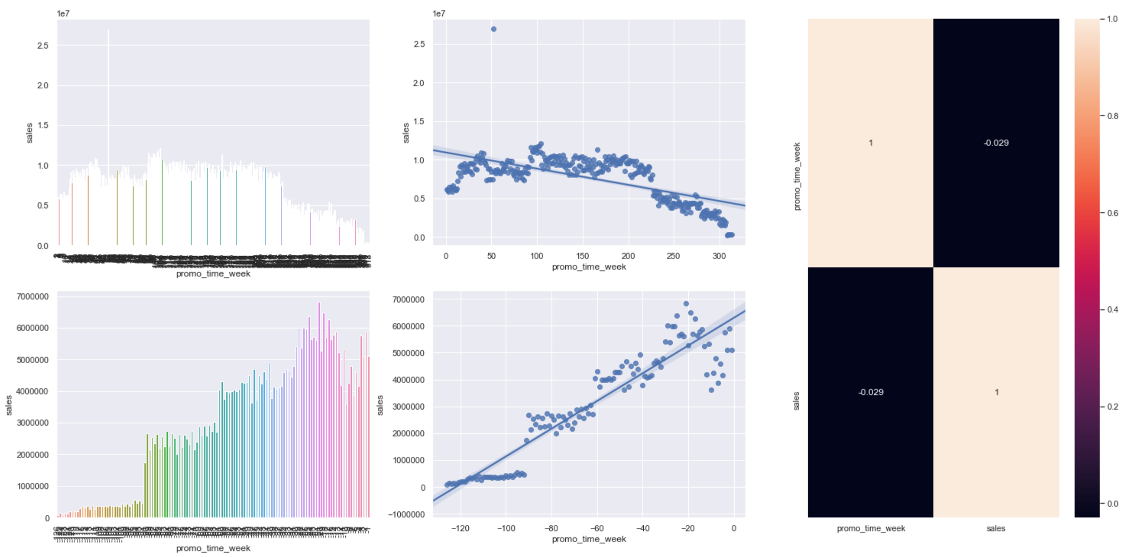
[31]:
```
plt.subplot( 1, 3, 1 )
aux1 = df4[['competition_time_month', 'sales']].groupby(␣
↪'competition_time_month' ).sum().reset_index()
aux2 = aux1[( aux1['competition_time_month'] < 120 ) & (␣
↪aux1['competition_time_month'] != 0 )]
sns.barplot( x='competition_time_month', y='sales', data=aux2 );
plt.xticks( rotation=90 );

plt.subplot( 1, 3, 2 )
sns.regplot( x='competition_time_month', y='sales', data=aux2 );

plt.subplot( 1, 3, 3 )
x = sns.heatmap( aux1.corr( method='pearson'), annot=True );
bottom, top = x.get_ylim()
x.set_ylim( bottom+0.5, top-0.5);
```



5.2.4 H4. Lojas com promoções ativas por mais tempo deveriam vender mais.

FALSA Lojas com promocoes ativas por mais tempo vendem menos, depois de um certo periodo de promocao

[32]:
```
aux1 = df4[['promo_time_week', 'sales']].groupby( 'promo_time_week').sum().
 ↪reset_index()

grid = GridSpec( 2, 3 )

plt.subplot( grid[0,0] )
aux2 = aux1[aux1['promo_time_week'] > 0] # promo extendido
sns.barplot( x='promo_time_week', y='sales', data=aux2 );
plt.xticks( rotation=90 );

plt.subplot( grid[0,1] )
sns.regplot( x='promo_time_week', y='sales', data=aux2 );

plt.subplot( grid[1,0] )
aux3 = aux1[aux1['promo_time_week'] < 0] # promo regular
sns.barplot( x='promo_time_week', y='sales', data=aux3 );
plt.xticks( rotation=90 );

plt.subplot( grid[1,1] )
sns.regplot( x='promo_time_week', y='sales', data=aux3 );

plt.subplot( grid[:,2] )
sns.heatmap( aux1.corr( method='pearson' ), annot=True );
```



5.2.5 H5. Lojas com mais dias de promoção deveriam vender mais.

5.2.6 H7. Lojas com mais promoções consecutivas deveriam vender mais.

FALSA Lojas com mais promocoes consecutivas vendem menos

[33]:
```python
df4[['promo', 'promo2', 'sales']].groupby( ['promo', 'promo2'] ).sum().reset_index()
```

[33]:

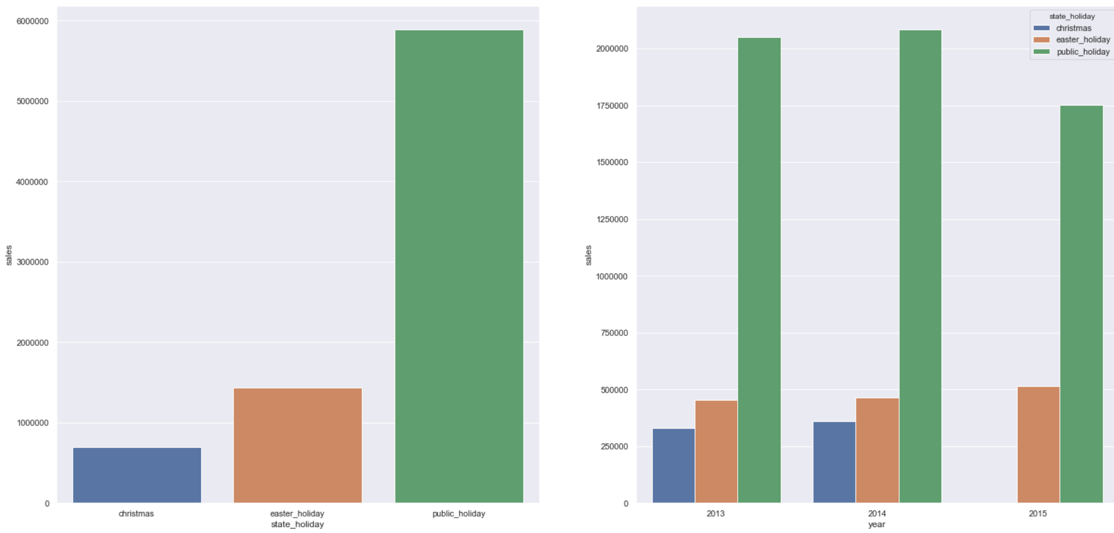
	promo	promo2	sales
0	0	0	1482612096
1	0	1	1289362241
2	1	0	1628930532
3	1	1	1472275754

[34]:
```python
aux1 = df4[( df4['promo'] == 1 ) & ( df4['promo2'] == 1 )][['year_week', 'sales']].groupby( 'year_week' ).sum().reset_index()
ax = aux1.plot()

aux2 = df4[( df4['promo'] == 1 ) & ( df4['promo2'] == 0 )][['year_week', 'sales']].groupby( 'year_week' ).sum().reset_index()
aux2.plot( ax=ax )

ax.legend( labels=['Tradicional & Extendida', 'Extendida']);
```

5.2.7 H8. Lojas abertas durante o feriado de Natal deveriam vender mais.

FALSA Lojas abertas durante o feriado do Natal vendem menos.

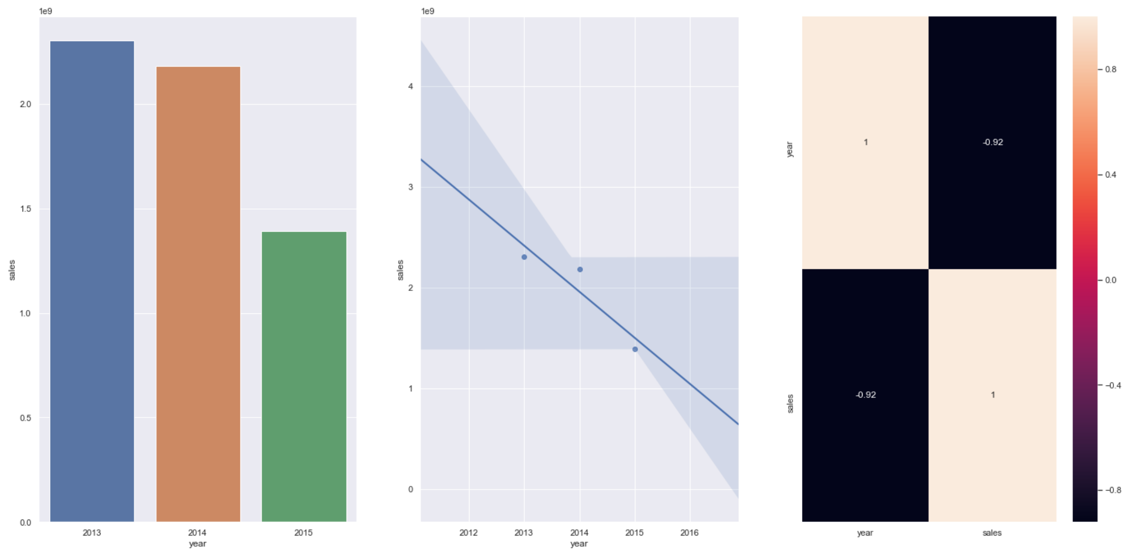
[35]:
```
aux = df4[df4['state_holiday'] != 'regular_day']

plt.subplot( 1, 2, 1 )
aux1 = aux[['state_holiday', 'sales']].groupby( 'state_holiday' ).sum().reset_index()
sns.barplot( x='state_holiday', y='sales', data=aux1 );

plt.subplot( 1, 2, 2 )
aux2 = aux[['year', 'state_holiday', 'sales']].groupby( ['year', 'state_holiday'] ).sum().reset_index()
sns.barplot( x='year', y='sales', hue='state_holiday', data=aux2 );
```



5.2.8 H9. Lojas deveriam vender mais ao longo dos anos.

FALSA Lojas vendem menos ao longo dos anos

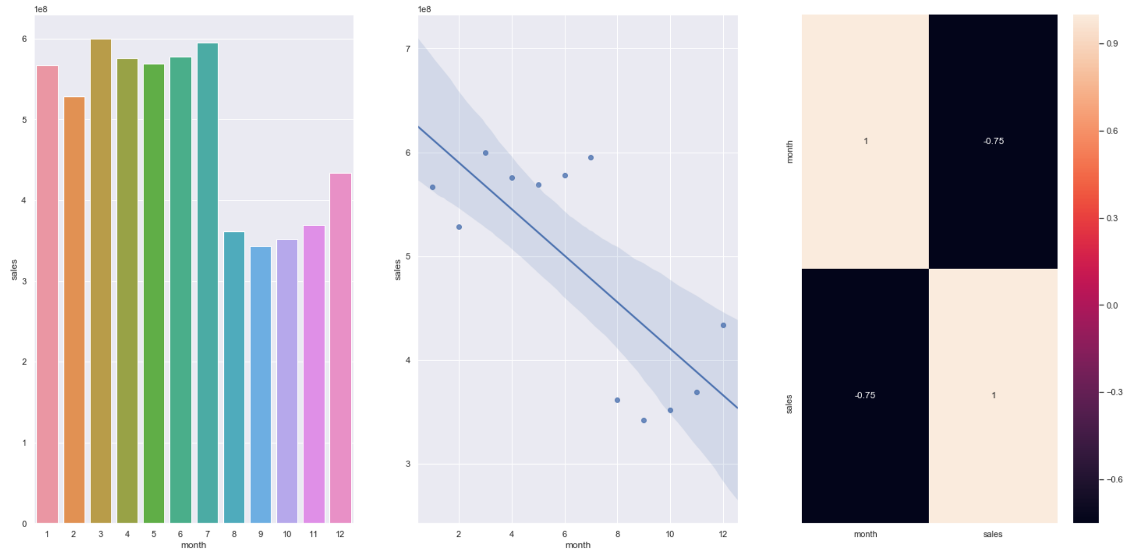
[36]:
```
aux1 = df4[['year', 'sales']].groupby( 'year' ).sum().reset_index()

plt.subplot( 1, 3, 1 )
sns.barplot( x='year', y='sales', data=aux1 );

plt.subplot( 1, 3, 2 )
sns.regplot( x='year', y='sales', data=aux1 );

plt.subplot( 1, 3, 3 )
sns.heatmap( aux1.corr( method='pearson' ), annot=True );
```



5.2.9 H10. Lojas deveriam vender mais no segundo semestre do ano.

FALSA Lojas vendem menos no segundo semestre do ano

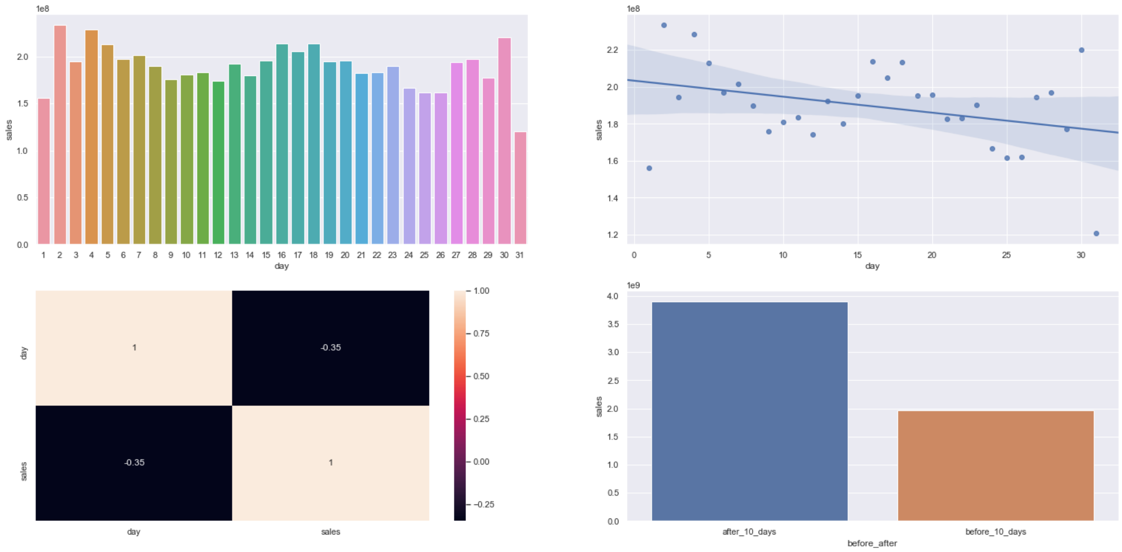
[37]:
```
aux1 = df4[['month', 'sales']].groupby( 'month' ).sum().reset_index()

plt.subplot( 1, 3, 1 )
sns.barplot( x='month', y='sales', data=aux1 );

plt.subplot( 1, 3, 2 )
sns.regplot( x='month', y='sales', data=aux1 );

plt.subplot( 1, 3, 3 )
sns.heatmap( aux1.corr( method='pearson' ), annot=True );
```



5.2.10 H11. Lojas deveriam vender mais depois do dia 10 de cada mês.

VERDADEIRA Lojas vendem mais depois do dia 10 de cada mes.

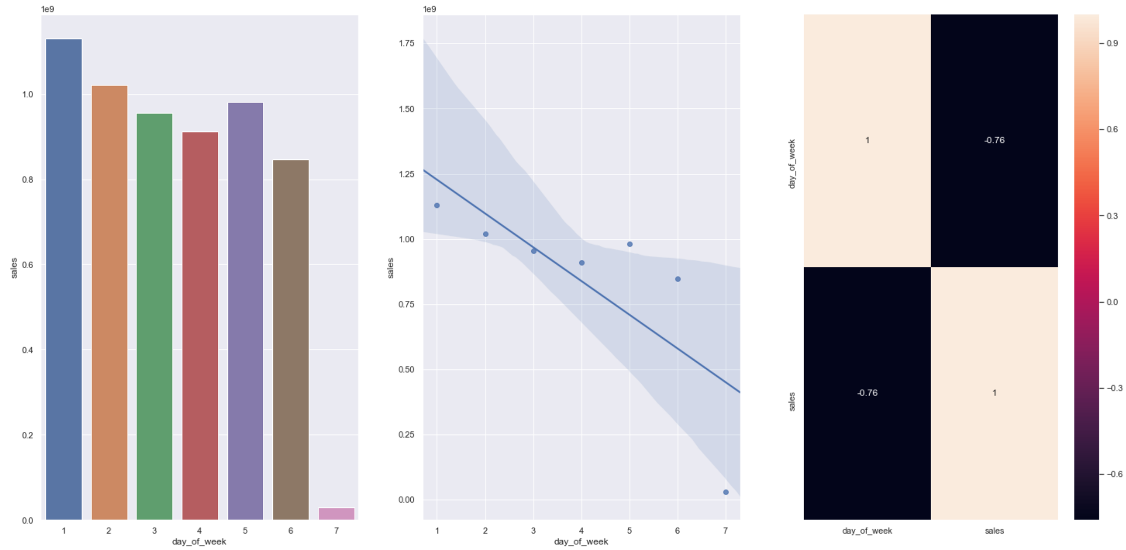
[38]:
```
aux1 = df4[['day', 'sales']].groupby( 'day' ).sum().reset_index()

plt.subplot( 2, 2, 1 )
sns.barplot( x='day', y='sales', data=aux1 );

plt.subplot( 2, 2, 2 )
sns.regplot( x='day', y='sales', data=aux1 );

plt.subplot( 2, 2, 3 )
sns.heatmap( aux1.corr( method='pearson' ), annot=True );

aux1['before_after'] = aux1['day'].apply( lambda x: 'before_10_days' if x <= 10␣
↪else 'after_10_days' )
aux2 =aux1[['before_after', 'sales']].groupby( 'before_after' ).sum().
↪reset_index()

plt.subplot( 2, 2, 4 )
sns.barplot( x='before_after', y='sales', data=aux2 );
```



5.2.11 H12. Lojas deveriam vender menos aos finais de semana.

VERDADEIRA Lojas vendem menos nos final de semana

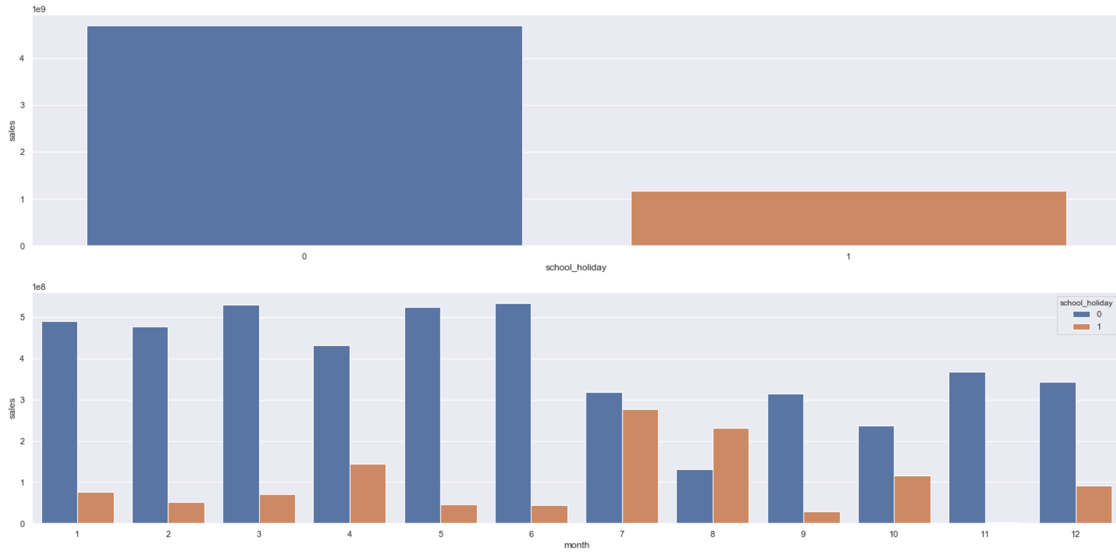
[39]:
```
aux1 = df4[['day_of_week', 'sales']].groupby( 'day_of_week' ).sum().
 ↪reset_index()

plt.subplot( 1, 3, 1 )
sns.barplot( x='day_of_week', y='sales', data=aux1 );

plt.subplot( 1, 3, 2 )
sns.regplot( x='day_of_week', y='sales', data=aux1 );

plt.subplot( 1, 3, 3 )
sns.heatmap( aux1.corr( method='pearson' ), annot=True );
```



5.2.12 H13. Lojas deveriam vender menos durante os feriados escolares.

VERDADEIRA Lojas vendem menos durante os feriadso escolares, except os meses de Julho e Agosto.

[40]:
```
aux1 = df4[['school_holiday', 'sales']].groupby( 'school_holiday' ).sum().
 ↪reset_index()
plt.subplot( 2, 1, 1 )
sns.barplot( x='school_holiday', y='sales', data=aux1 );

aux2 = df4[['month', 'school_holiday', 'sales']].groupby(␣
 ↪['month','school_holiday'] ).sum().reset_index()
plt.subplot( 2, 1, 2 )
sns.barplot( x='month', y='sales', hue='school_holiday', data=aux2 );
```



5.2.13 4.2.1. Resumo das Hipoteses

[41]:
```python
from tabulate import tabulate
```

[42]:
```python
tab =[['Hipoteses', 'Conclusao', 'Relevancia'],
      ['H1', 'Falsa', 'Baixa'],
      ['H2', 'Falsa', 'Media'],
      ['H3', 'Falsa', 'Media'],
      ['H4', 'Falsa', 'Baixa'],
      ['H5', '-', '-'],
      ['H7', 'Falsa', 'Baixa'],
      ['H8', 'Falsa', 'Media'],
      ['H9', 'Falsa', 'Alta'],
      ['H10', 'Falsa', 'Alta'],
      ['H11', 'Verdadeira', 'Alta'],
      ['H12', 'Verdadeira', 'Alta'],
      ['H13', 'Verdadeira', 'Baixa'],
     ]
print( tabulate( tab, headers='firstrow' ) )
```

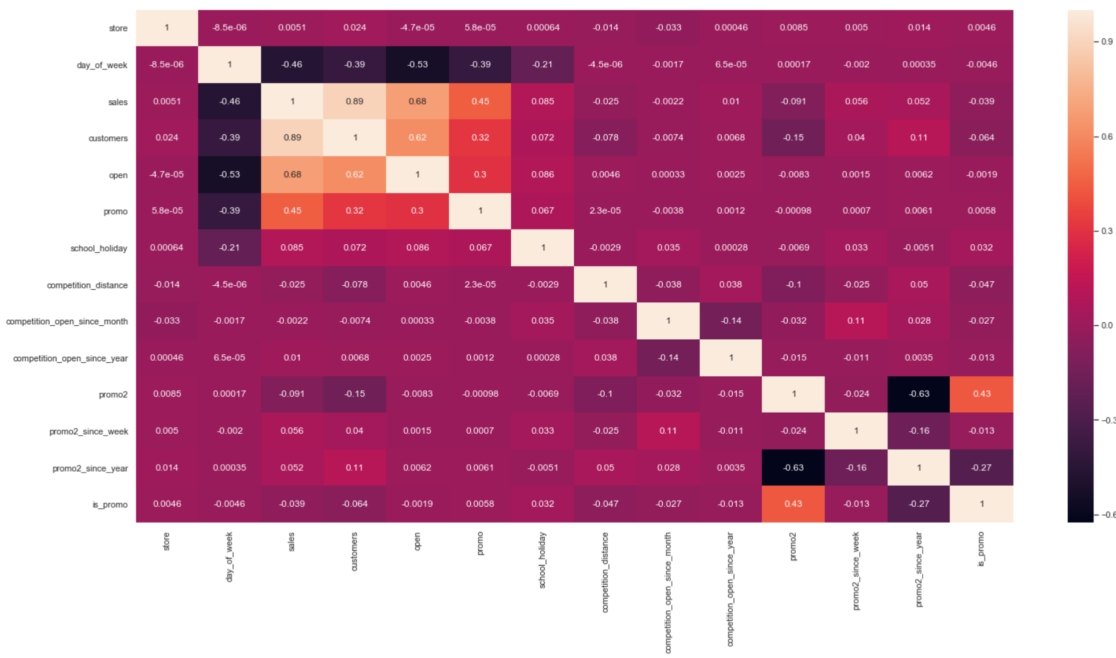
```
Hipoteses    Conclusao    Relevancia
-----------  -----------  ------------
H1           Falsa        Baixa
H2           Falsa        Media
H3           Falsa        Media
H4           Falsa        Baixa
H5           -            -
H7           Falsa        Baixa
H8           Falsa        Media
```

```
H9           Falsa        Alta
H10          Falsa        Alta
H11          Verdadeira   Alta
H12          Verdadeira   Alta
H13          Verdadeira   Baixa
```

5.3 4.3. Analise Multivariada

5.3.1 4.3.1. Numerical Attributes

[43]:
```python
correlation = num_attributes.corr( method='pearson' )
sns.heatmap( correlation, annot=True );
```



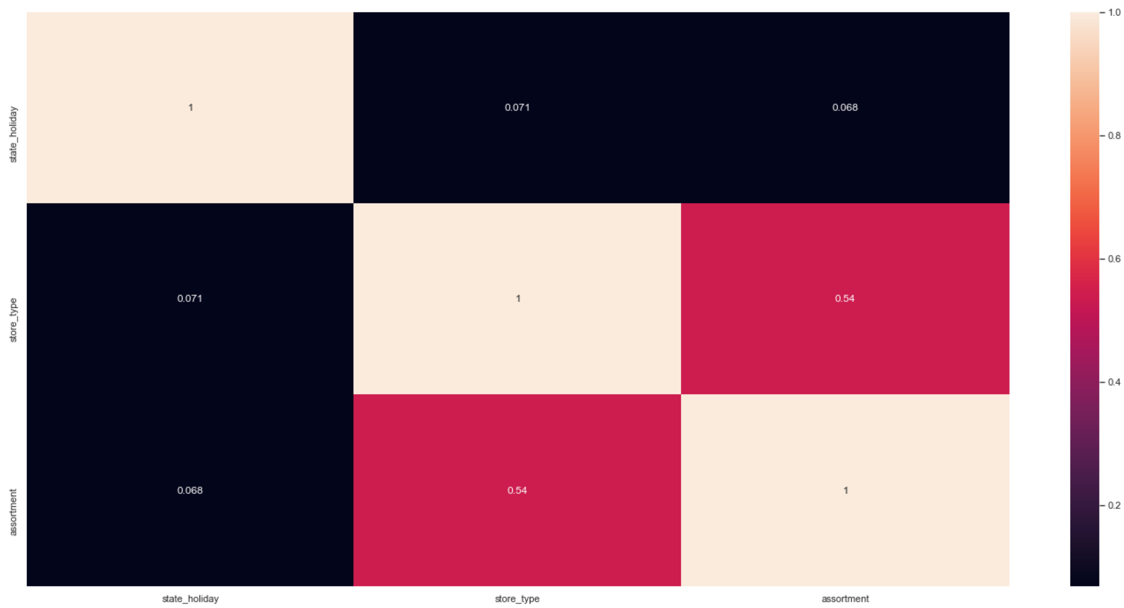
5.3.2 4.3.2. Categorical Attributes

[44]:
```python
# only categorical data
a = df4.select_dtypes( include='object' )

# Calculate cramer V
a1 = cramer_v( a['state_holiday'], a['state_holiday'] )
a2 = cramer_v( a['state_holiday'], a['store_type'] )
a3 = cramer_v( a['state_holiday'], a['assortment'] )

a4 = cramer_v( a['store_type'], a['state_holiday'] )
a5 = cramer_v( a['store_type'], a['store_type'] )
a6 = cramer_v( a['store_type'], a['assortment'] )
```

```
a7 = cramer_v( a['assortment'], a['state_holiday'] )
a8 = cramer_v( a['assortment'], a['store_type'] )
a9 = cramer_v( a['assortment'], a['assortment'] )

# Final dataset
d = pd.DataFrame( {'state_holiday': [a1, a2, a3],
               'store_type': [a4, a5, a6],
               'assortment': [a7, a8, a9]  })
d = d.set_index( d.columns )

sns.heatmap( d, annot=True )
```

```
<ipython-input-2-a3b24802d76f>:2: FutureWarning: Method .as_matrix will be
removed in a future version. Use .values instead.
  cm = pd.crosstab( x, y ).as_matrix()
```

[44]: <matplotlib.axes._subplots.AxesSubplot at 0x11c2eb130>



[ ]:

6 5.0. PASSO 05 - DATA PREPARATION

[87]:
```
df5 = df4.copy()
```

6.1 5.1. Normalizacao

[ ]:

6.2 5.2. Rescaling

[88]:
```
rs = RobustScaler()
mms = MinMaxScaler()

# competition distance
df5['competition_distance'] = rs.fit_transform( df5[['competition_distance']].
 ↪values )

# competition time month
df5['competition_time_month'] = rs.fit_transform(␣
 ↪df5[['competition_time_month']].values )

# promo time week
df5['promo_time_week'] = mms.fit_transform( df5[['promo_time_week']].values )

# year
df5['year'] = mms.fit_transform( df5[['year']].values )
```

6.3 5.3. Transformacao

6.3.1 5.3.1. Encoding

[89]:
```
# state_holiday - One Hot Encoding
df5 = pd.get_dummies( df5, prefix=['state_holiday'], columns=['state_holiday'] )

# store_type - Label Encoding
le = LabelEncoder()
df5['store_type'] = le.fit_transform( df5['store_type'] )

# assortment - Ordinal Encoding
assortment_dict = {'basic': 1,  'extra': 2, 'extended': 3}
df5['assortment'] = df5['assortment'].map( assortment_dict )
```

6.3.2 5.3.2. Response Variable Transformation

[90]:
```
df5['sales'] = np.log1p( df5['sales'] )
```

6.3.3 5.3.2. Nature Transformation

[91]:
```
# day of week
df5['day_of_week_sin'] = df5['day_of_week'].apply( lambda x: np.sin( x * ( 2. *␣
 ↪np.pi/7 ) ) )
```

```
df5['day_of_week_cos'] = df5['day_of_week'].apply( lambda x: np.cos( x * ( 2. *␣
↪np.pi/7 ) ) )

# month
df5['month_sin'] = df5['month'].apply( lambda x: np.sin( x * ( 2. * np.pi/12 )␣
↪) )
df5['month_cos'] = df5['month'].apply( lambda x: np.cos( x * ( 2. * np.pi/12 )␣
↪) )

# day
df5['day_sin'] = df5['day'].apply( lambda x: np.sin( x * ( 2. * np.pi/30 ) ) )
df5['day_cos'] = df5['day'].apply( lambda x: np.cos( x * ( 2. * np.pi/30 ) ) )

# week of year
df5['week_of_year_sin'] = df5['week_of_year'].apply( lambda x: np.sin( x * ( 2.␣
↪* np.pi/52 ) ) )
df5['week_of_year_cos'] = df5['week_of_year'].apply( lambda x: np.cos( x * ( 2.␣
↪* np.pi/52 ) ) )
```

[92]:
```
df5.head()
```

[92]:
```
   store  day_of_week       date     sales  promo  school_holiday  store_type  \
assortment  competition_distance  competition_open_since_month  \
competition_open_since_year  promo2  promo2_since_week  promo2_since_year  \
is_promo  year  month  day  week_of_year year_week competition_since  \
competition_time_month promo_since  promo_time_week  state_holiday_christmas  \
state_holiday_easter_holiday  state_holiday_public_holiday  \
state_holiday_regular_day  day_of_week_sin  day_of_week_cos  month_sin  \
month_cos   day_sin   day_cos  week_of_year_sin  week_of_year_cos
0      1            5 2015-07-31  8.568646      1               1           2
1             -0.170968                             9
2008        0                31                2015          0   1.0      7   31
31   2015-30        2008-09-01                0.918919  2015-07-27
0.287016                      0                              0
0                          1        -0.974928        -0.222521       -0.5
-0.866025  0.207912  0.978148         -0.568065         -0.822984
1      2            5 2015-07-31  8.710290      1               1           0
1             -0.283871                            11
2007        1                13                2010          1   1.0      7   31
31   2015-30        2007-11-01                1.054054  2010-03-22
0.922551                      0                              0
0                          1        -0.974928        -0.222521       -0.5
-0.866025  0.207912  0.978148         -0.568065         -0.822984
2      3            5 2015-07-31  9.025816      1               1           0
1              1.903226                            12
2006        1                14                2011          1   1.0      7   31
31   2015-30        2006-12-01                1.202703  2011-03-28
```

```
0.801822                       0                            0
0                          1        -0.974928        -0.222521       -0.5
-0.866025  0.207912  0.978148         -0.568065         -0.822984
3      4           5 2015-07-31  9.546527      1              1           2
3             -0.275806                            9
2009       0                31               2015         0   1.0      7   31
31   2015-30        2009-09-01               0.743243  2015-07-27
0.287016                       0                            0
0                          1        -0.974928        -0.222521       -0.5
-0.866025  0.207912  0.978148         -0.568065         -0.822984
4      5           5 2015-07-31  8.481151      1              1           0
1              4.448387                            4
2015       0                31               2015         0   1.0      7   31
31   2015-30        2015-04-01              -0.162162  2015-07-27
0.287016                       0                            0
0                          1        -0.974928        -0.222521       -0.5
-0.866025  0.207912  0.978148         -0.568065         -0.822984
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: