Projeto - Previsão de receita

Contexto:

O contexto não foi definido. Entretanto, para contextualizar o projeto **estarei adotando a seguinte abordagem:**

Contexto: Uma empresa chamada QualityShoes opera no ramo de e-commerce dentro da área de artigos esportivos. Como a empresa ainda não tem um time de dados estruturado, ela fez um contrato com uma empresa de consultoria especializada no ramo de transformação digital. Com esse contrato, fui chamado para atuar como cientista de dados. Comecei a marcar algumas reuniões com as áreas de negócio da empresa cliente, e após uma primeira reunião com o CFO da quality e o nosso time de dados, constatamos que o primeiro projeto a ser realizado deveria ser a projeção de receita bruta/líquida. O portfólio da quality é bem amplo, com mais de 100 mil clientes atacadistas e varejistas. A ideia inicial do projeto é entender as principais variáveis que estão correlacionadas à receita e retirar insights importantes para o time de negócio.

Objetivo de negócio:

- Projeção de receita para os próximos 6 meses

Entendimento do negócio:

1. Qual a motivação:
   - A projeção de receita nos próximos 6 meses surgiu a partir da necessidade de desenhar o melhor budget para investimentos nas áreas da empresa, quanto mais seguro é a minha predição receita, menos riscos a empresa estará correndo com investimentos mais arriscados.
2. Qual a causa raíz do problema:
   - Dificuldade em determinar o melhor budget para investimentos internos.
3. Quem é o dono do problema:
   - Diretor financeiro (CFO) da QualityShoes
4. Qual é o formato da solução?
   - **Granularidade:** Previsão de receita diária nos próximos 183 dias (6 meses)
   - **Tipo de problema:** Previsão de receita (Regressão)
   - **Potenciais métodos:** Séries temporais e regressão com algumas modificações
   - **Formato de entrega:**
     - O valor total da receita líquida no final dos 6 meses.
     - A entrega será pelo app do streamlit
     - Checagem trimestral

0.0 Imports

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import os
import pathlib
import datetime

from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.arima.model import ARIMA

from IPython.core.display  import HTML
from IPython.display       import Image

import warnings
warnings.filterwarnings("ignore")
```

0.1 Helper Functions

```
def jupyter_settings():
    %matplotlib inline
    plt.style.use( 'bmh' )
    plt.rcParams['figure.figsize'] = [15, 10]
    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()
jupyter_settings()
```

```
<IPython.core.display.HTML object>
```

```
# Função que extrai as letras do texto
def _extract_letter(text):
    text = re.sub('[A-Z]','',text)
    return text

# Função que define o MAPE
def mean_absolute_percentage_error(y,yhat):
    mape = np.mean(np.abs((y - yhat)/y))
    return mape
```

```
# Função para análise de métricas de regressão
def ml_error(name_model, y, yhat):
    mae = mean_absolute_error(y,yhat)
    mape = mean_absolute_percentage_error(y,yhat)
    rmse = np.sqrt(mean_squared_error(y,yhat))

    return pd.DataFrame({'Model Name': name_model,
                         'MAE': mae, #Erro médio absoluto
                         'MAPE': mape,  # Percentual do erro médio
absoluto
                         'RMSE': rmse}, index=[0]) # Raíz quadrada do
erro médio
```

0.2 Variables path

```
PATH_ROOT = pathlib.Path('..').resolve()
DATA_ROOT = os.path.join(PATH_ROOT,'data')
DATA_RAW = os.path.join(DATA_ROOT,'raw')
DATA_PREPROCESSED = os.path.join(DATA_RAW, 'preprocessed')

DOC_ROOT = os.path.join(PATH_ROOT, 'doc')
IMAGE_PATH = os.path.join(DOC_ROOT,'image')
# File data
_data_raw = os.path.join(DATA_RAW, 'Dataset_teste_Just_BI.csv')
_image_map_hypothesis = os.path.join(IMAGE_PATH,'map_hypothesis.png')
```

0.3 Data Loading

```
df = pd.read_csv(_data_raw, sep=';')
```

1.0 Data Description

```
df1 = df.copy()

df1.head()
```

```
   customer_id customer_acquisition_channel  year week  net_revenue
gross_revenue  boxes
0       206461               Paid Marketing  2014  W09           71
71      2
1       462640               Paid Marketing  2015  W25           28
56      1
2       666461                     Referral  2015  W50           40
40      1
3       183202                     Referral  2013  W42           18
37      1
4       410993                     Referral  2014  W29            0
37      1
```

1.1 Data info General

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 715875 entries, 0 to 715874
Data columns (total 7 columns):
 #   Column                        Non-Null Count   Dtype 
---  ------                        --------------   ----- 
 0   customer_id                   715875 non-null  int64 
 1   customer_acquisition_channel  715875 non-null  object
 2   year                          715875 non-null  int64 
 3   week                          715875 non-null  object
 4   net_revenue                   715875 non-null  int64 
 5   gross_revenue                 715875 non-null  int64 
 6   boxes                         715875 non-null  int64 
dtypes: int64(5), object(2)
memory usage: 38.2+ MB
```

Atributos:

- Id do Cliente
- Canal de aquisição do cliente
- Ano da venda
- Semana da venda
- Receita líquida
- Receita bruta
- Caixas

Obs: Não temos dados faltantes

1.2 Check NA

```
# Percentual dos dados faltantes
(df1.isna().sum() / len(df1))*100
```

```
customer_id                     0.0
customer_acquisition_channel    0.0
year                            0.0
week                            0.0
net_revenue                     0.0
gross_revenue                   0.0
boxes                           0.0
dtype: float64
```

1.3 Descriptive Statistics

Change DTypes

```
# Extraindo os números da semana
df1['weekofyear_num'] = df1['week'].apply(_extract_letter)

df1['weekofyear_num']=df1['weekofyear_num'].astype('int64')
df1['net_revenue']= df1['net_revenue'].astype('float64')
df1['gross_revenue'] = df1['gross_revenue'].astype('float64')
```

```
numerical_attributes = df1.select_dtypes(include=['int64','float64'])
categorical_attributes =
df1.select_dtypes(exclude=['int64','float64','datetime64'])
```

1.3.1 Numerical Attributes

```
numerical_attributes.describe().T
```

```
                   count           mean             std     min
25%       50%       75%       max
customer_id     715875.0  323664.862498  186136.720066   103.0
159325.0  289541.0  476431.0  746721.0
year            715875.0    2014.196999       0.754568  2013.0
2014.0    2014.0    2015.0    2015.0
net_revenue     715875.0      36.415437      13.646022     0.0
37.0      37.0      37.0     532.0
gross_revenue   715875.0      41.430585       9.752502    14.0
37.0      37.0      40.0     532.0
boxes           715875.0       1.034697       0.190199     1.0
1.0       1.0       1.0      17.0
weekofyear_num  715875.0      27.491899      15.105624     1.0
14.0      28.0      41.0      53.0
```

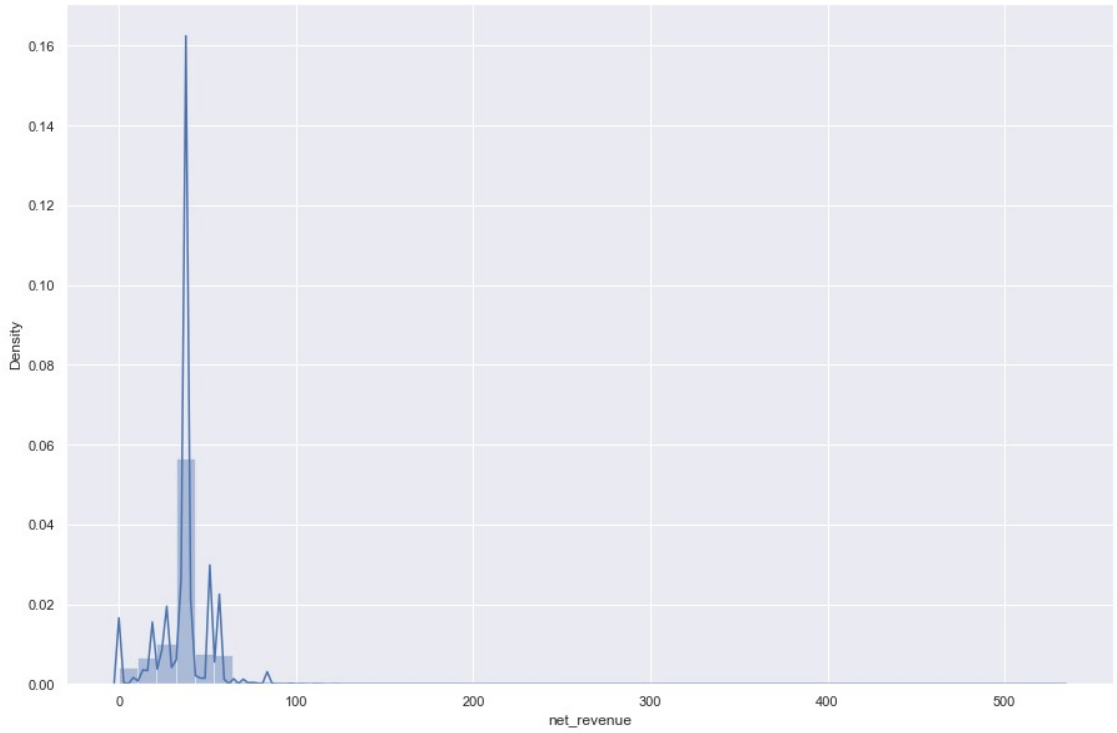
```
# Central Tendency - Mean, Median (Métricas que resume a
representatividade dos dados)
# Dispersion - std, min, max, range, skew, kurtosis (Medidas de
dispersão em relação a média)
# skew : Como é a deformação da distribuição em relação a normal, se a
deformação for mais para a direita teremos uma skew positiva, se for
mais para a esquerda teremos uma skew negativa.
# Kurtosis: Métrica em relação a concentração dos dados, quanto maior
a Curtose positiva, maior será o pico de concentração dos dados.
Entretanto, quanto menor a Curtose, mais dispersos serão os dados,
achatando ainda mais minha curva de distribuição.
skew = pd.DataFrame(numerical_attributes.apply( lambda x: x.skew() ))
kurtosis = pd.DataFrame(numerical_attributes.apply( lambda x:
x.kurtosis() ))
range = pd.DataFrame(numerical_attributes.apply( lambda x: x.max() -
x.min() ))
describe = pd.DataFrame(numerical_attributes.describe()).T
describe['range'] = range
describe['skew'] = skew
describe['kurtosis'] = kurtosis
describe = describe.reset_index()
describe = describe.rename(columns={'index':'Attributes'})
describe
```

```
       Attributes     count           mean             std     min
25%       50%       75%       max     range      skew    kurtosis
0     customer_id  715875.0  323664.862498  186136.720066   103.0
159325.0  289541.0  476431.0  746721.0  746618.0  0.403586   -1.048270
1            year  715875.0    2014.196999       0.754568  2013.0
```

```
2014.0    2014.0    2015.0    2015.0       2.0 -0.342490   -1.179912
2     net_revenue  715875.0      36.415437      13.646022     0.0
37.0      37.0      37.0     532.0     532.0  0.526808   10.403764
3   gross_revenue  715875.0      41.430585       9.752502    14.0
37.0      37.0      40.0     532.0     518.0  3.435058   36.807084
4           boxes  715875.0       1.034697       0.190199     1.0
1.0       1.0       1.0      17.0      16.0  6.621214  112.525487
5  weekofyear_num  715875.0      27.491899      15.105624     1.0
14.0      28.0      41.0      53.0      52.0 -0.061525   -1.238845
```

```
sns.distplot(df1['net_revenue'])
```

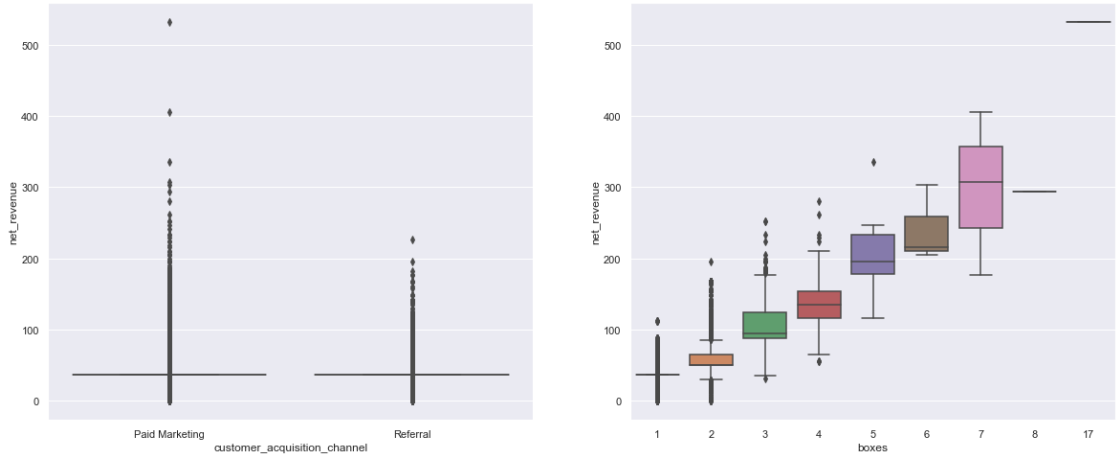
```
<AxesSubplot:xlabel='net_revenue', ylabel='Density'>
```



1.3.2 Categorical Attributes

```
categorical_attributes['boxes'] = numerical_attributes['boxes']
categorical_attributes = categorical_attributes.drop(['week'], axis=1)
#categorical_attributes.apply(lambda x: x.unique().shape[0])

plt.figure(figsize = [20,8])
plt.subplot(1,2,1)
sns.boxplot(x='customer_acquisition_channel' , y= 'net_revenue' ,
data= df1)

plt.subplot(1,2,2)
sns.boxplot(x='boxes' , y= 'net_revenue' , data= df1)
```

```
<AxesSubplot:xlabel='boxes', ylabel='net_revenue'>
```



Obs:

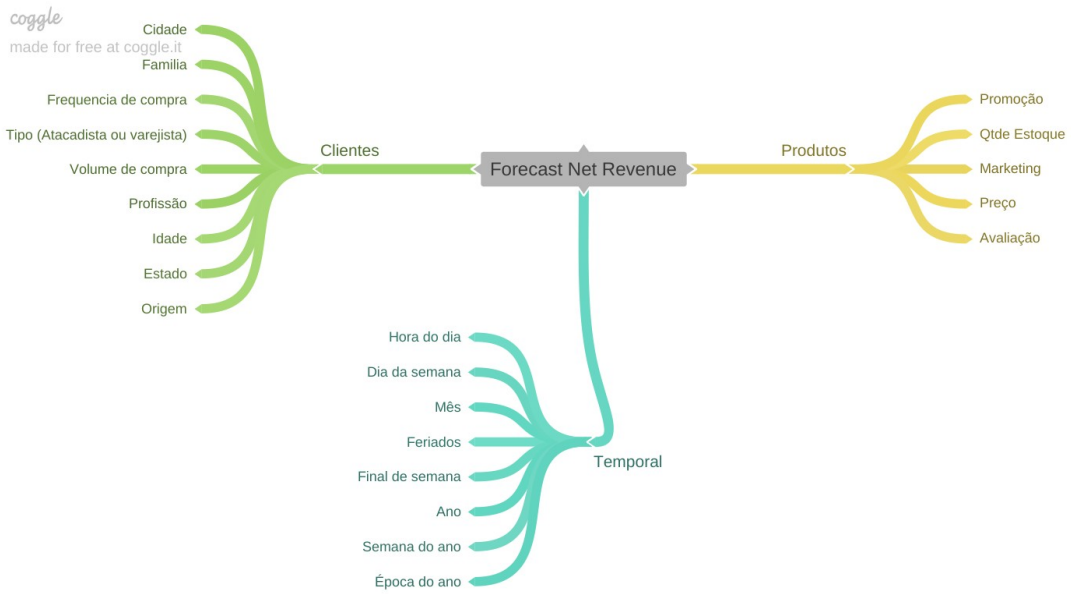
- Ao comparar os tipos de canal para aquisição dos clientes é perceptível que o marketing pago tem levemente uma distribuição de receita líquida mais favorecida que o canal de referências. Mas vale ressaltar que isso vale apenas para pontos extremos (Outliers), a média são equivalentes entre os dois tipos de acordo com o boxplot.
- O Box 7 tem uma melhor arrecadação de receita líquida quando comparado com os demais boxes (Exceto o box 17, mas como não existe uma boa variância, pode ser algo já previsível na receita). Além de não apresentar outliers, a média do box 7 está acima da mediana, então considera-se que temos um bom desempenho de arrecadação de receita líquida para esse box.

2.0 Feature Engineering

```
df2 = df1.copy()
```

2.1 Mind Map Hypothesis

```
Image(_image_map_hypothesis)
```



2.2 Create Hypothesis

Sem hipóteses para esse primeiro ciclo

2.3 Feature Engineering

```
# Substituindo W53 por W52 e considerando que os anos tenham apenas 52 semanas e o mês 4 semanas normais.
df2['week'] = df2['week'].str.replace('W53','W52')

# Obtendo o campo data considerando que para cada semana do ano, o dia da semana adotado será segunda-feira.
df2['weekofyear'] = df2.apply(lambda x: str(x['year'])+'-'+x['week'], axis=1)
df2['date'] = df2.apply(lambda x: datetime.datetime.strptime(x['weekofyear']+ '-1', "%Y-W%W-%w" ), axis=1)

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

# Deduções: Impostos, despesas, cancelamentos e etc...

df2['deductions'] = df2['gross_revenue'] - df2['net_revenue']
```

3.0 Filtering variables

Nessa etapa estaremos filtrando as variáveis que não se adequam ao modelo em produção ou alguns registros que podem não fazer sentido.

```
df3 = df2.copy()

df3.head(3).T
```

```
                                                0                    1
2
customer_id                                206461               462640
666461
customer_acquisition_channel       Paid Marketing       Paid Marketing
Referral
year                                         2014                 2015
2015
week                                          W09                  W25
W50
net_revenue                                  71.0                 28.0
40.0
gross_revenue                                71.0                 56.0
40.0
boxes                                           2                    1
1
weekofyear_num                                  9                   25
50
weekofyear                                2014-W09             2015-W25
2015-W50
date                           2014-03-03 00:00:00  2015-06-22 00:00:00
2015-12-14 00:00:00
month                                           3                    6
12
deductions                                    0.0                 28.0
0.0
```

Não há nenhuma variável que possa ter algum tipo de restrição relacionado ao tipo de negócio. E em relação ao filtro de registros pode-se observar na seção 1.3.1 Numerical Attributes que não há valores negativos ou algum registro que esteja fora da normalidade.

4.0 Exploratory Data Analysis - EDA

Como as variáveis impactam o fenômeno? e qual a força desse impacto?

- Quais insigths podem ser retirados?

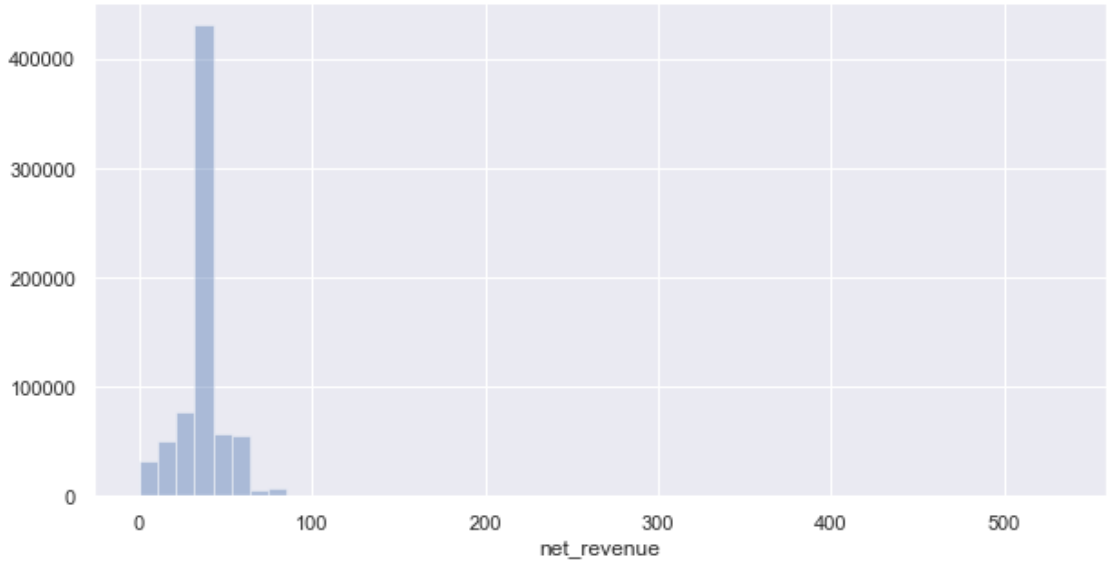
```
df4 = df3.copy()

num_attributes = df4.select_dtypes(include=['int64','float64'])
cat_attributes =
df4.select_dtypes(exclude=['int64','float64','datetime64[ns]'])
```

4.1 Análise Univariada

4.1.1 Target Variable

```
plt.figure(figsize=[10,5])
sns.distplot(df4['net_revenue'], kde=False)
```

```
<AxesSubplot:xlabel='net_revenue'>
```



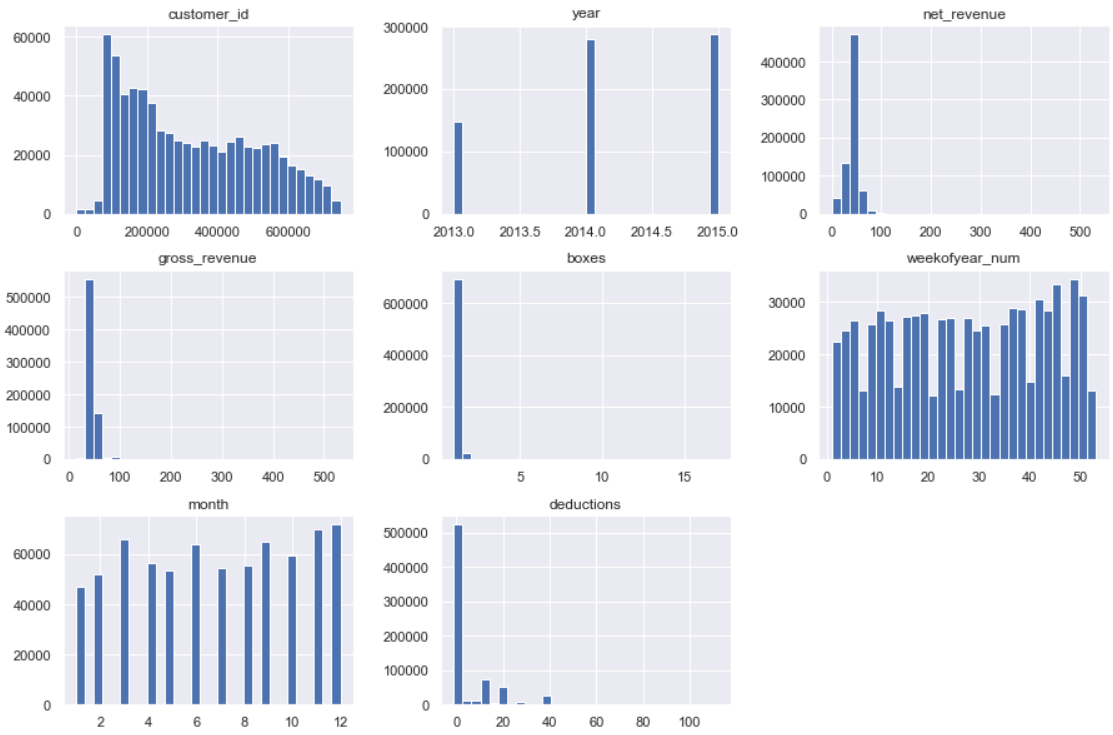
Obs: A maioria dos algoritmos de ML foram performados com algumas condições, e uma delas, normalmente, é a curva de distribuição da variável target, de forma que quanto mais se aproxima de uma variável normal, melhor é o seu resultado.

Por isso, precisamos transformar a variável target de forma que corresponda a uma curva normal ou pelo menos tenha uma aparência semelhante.

Existem várias técnicas de transformação, como por exemplo o uso do log.

4.1.2 Numerical Variable

```
num_attributes.hist(bins=30);
```



OBSERVAÇÕES:

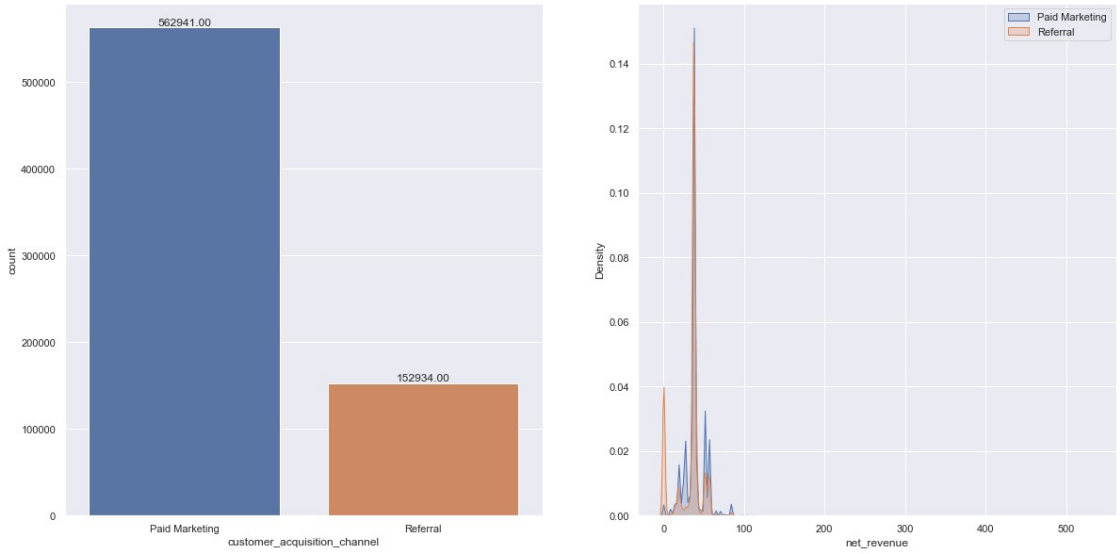
- ANO: O volume de compras aumentou ao longo dos anos
- RECEITA LÍQUIDA: Apesar de termos alguns outliers próximos a R$ 500,00, o volume de compras se concentra no valor de R$ 37,00.
- RECEITA BRUTA: A análise é semelhante a receita líquida, mas com uma média de R$ 47,00
- BOXES: Apesar de ter sido analisado na seção 1.3.2 (Data description) que os melhores resultados para receita líquida seriam dos boxes 7, 8 e 17, percebemos que o maior volume de compras está associado aos boxes 1 e 2. Nesse sentido, vale a pena entender os motivos de não serem vendidos itens dos boxes que tem maior concentração de receita e avaliar o custo-benefício de investimentos para vender mais essas categorias.
- SEMANAS: É observado que nas duas primeiras semanas há uma maior incidência de geração de receita do que nas semanas posteriores do mesmo mês.
- MESES: Aparentemente temos uma sazonalidade de 3 meses.
- DEDUÇÕES: As deduções sobre as vendas de qualquer tipo de natureza estão na média de R$ 5,00. Inclusive, seria legal observar o montante dessas deduções ao longo do tempo.

4.1.3 Categorical Variable

```
cat_attributes['customer_acquisition_channel'].value_counts()
```

```
Paid Marketing    562941
Referral          152934
Name: customer_acquisition_channel, dtype: int64
```

```
# Criar um grid com subplot
plt.figure(figsize=[20,10])
# State_holiday
plt.subplot(1,2,1)
a = df4.copy()
ax = sns.countplot(a['customer_acquisition_channel'])
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha = 'center',
                va = 'baseline',
                xytext = (0,2),
                textcoords = 'offset points')


plt.subplot(1,2,2)
sns.kdeplot(df4[df4['customer_acquisition_channel'] == 'Paid
Marketing']['net_revenue'],
            label = 'Paid Marketing', # Nome da linha
            shade = True, # Deixar mais transparente
            legend= True)

sns.kdeplot(df4[df4['customer_acquisition_channel'] == 'Referral']
['net_revenue'],
            label = 'Referral', # Nome da linha
            shade = True, # Deixar mais transparente
            legend= True)

plt.legend();
```



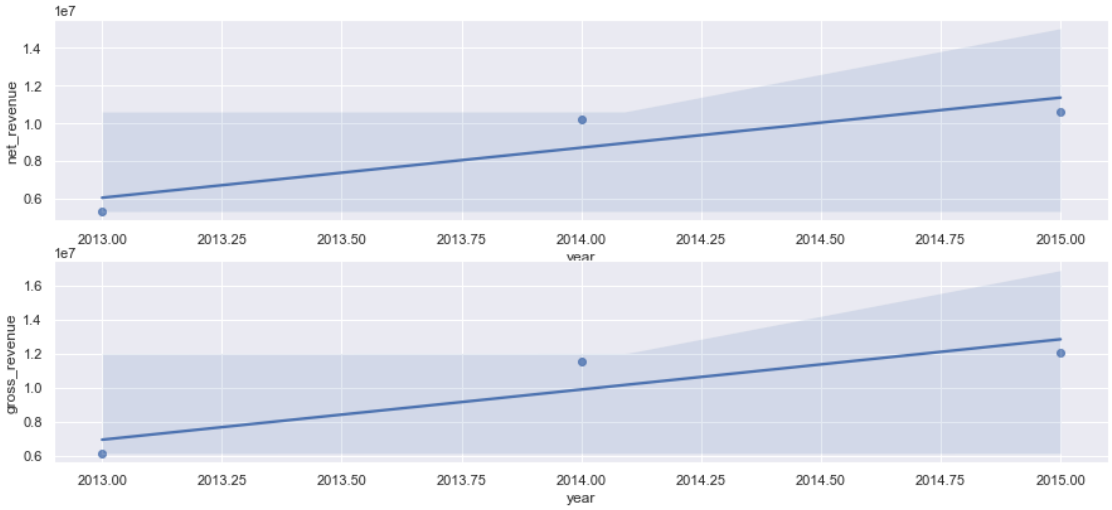
- É observado que temos uma correlação entre a receita líquida e o tipo de canal de aquisição, sendo que geralmente o marketing pago se sobressai em relação ao canal de referência.

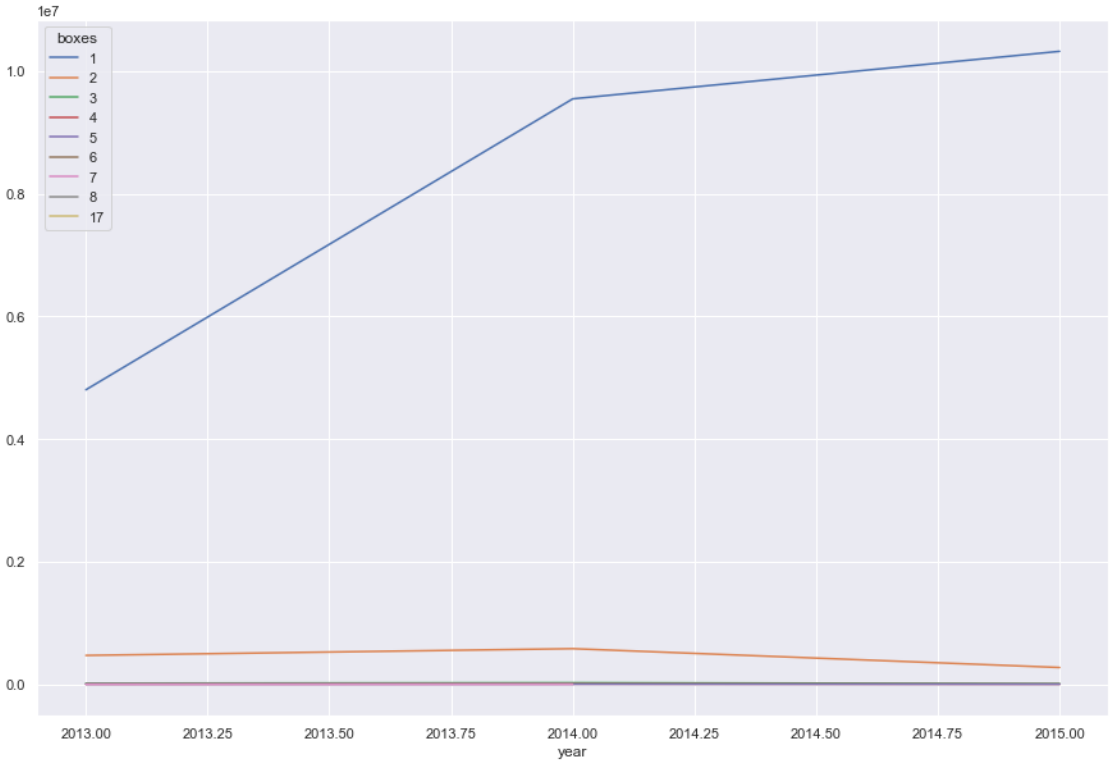
4.2 Análise bivariada

A1. Resultados de vendas por ano: Gross Revenue, Net Revenue, Boxes (em Gráfico e Tabela)

```
plt.subplot(3,1,1)
aux1 = df4[['net_revenue','gross_revenue',
'deductions','year']].groupby('year').sum().reset_index()
sns.regplot(x='year', y='net_revenue', data=aux1)

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

#plt.subplot(3,1,3)
aux2 =
df4[['year','boxes','net_revenue']].groupby(['year','boxes']).sum().re
set_index()
aux2.pivot( index= 'year', columns= 'boxes', values=
'net_revenue').plot()
```

```
<AxesSubplot:xlabel='year'>
```





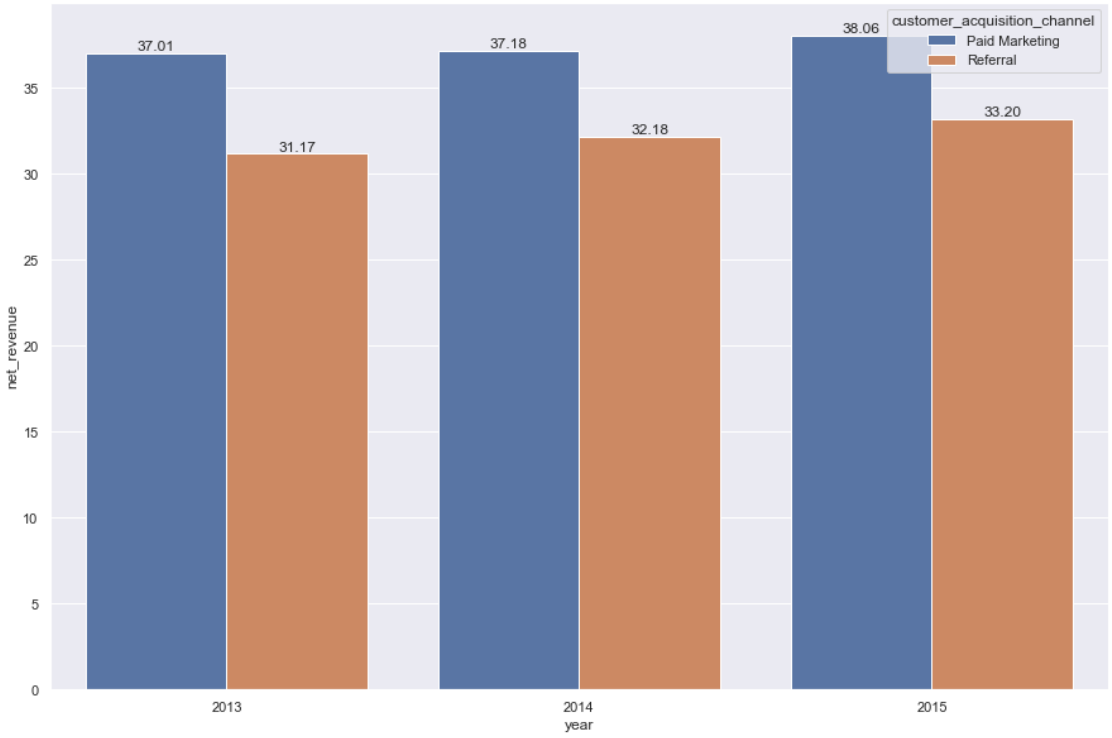
aux2

	year	boxes	net_revenue
0	2013	1	4803558.0
1	2013	2	470482.0
2	2013	3	17038.0
3	2013	4	2217.0
4	2013	5	1115.0
5	2013	6	303.0
6	2013	7	583.0
7	2013	17	532.0
8	2014	1	9549856.0
9	2014	2	580355.0
10	2014	3	25075.0
11	2014	4	4626.0
12	2014	5	946.0
13	2014	7	308.0
14	2014	8	294.0
15	2015	1	10325228.0
16	2015	2	273228.0
17	2015	3	10688.0
18	2015	4	1822.0
19	2015	5	226.0
20	2015	6	421.0

- O boxe 1 tem um aumento na receita ao longo dos anos, mas isso não é verdade em relação aos outros boxes.

A2. Qual customer_acquision_channel teve maior Ticket Médio em 2013 e em 2015?

```
aux1 =
df4[['net_revenue','customer_acquisition_channel','year']].groupby(['customer_acquisition_channel','year']).mean().reset_index()
cx = sns.barplot(x='year', y='net_revenue', data=aux1,
hue='customer_acquisition_channel')
for p in cx.patches:
    cx.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2, p.get_height()),
                ha = 'center',
                va = 'baseline',
                xytext = (0,2),
                textcoords = 'offset points')
```

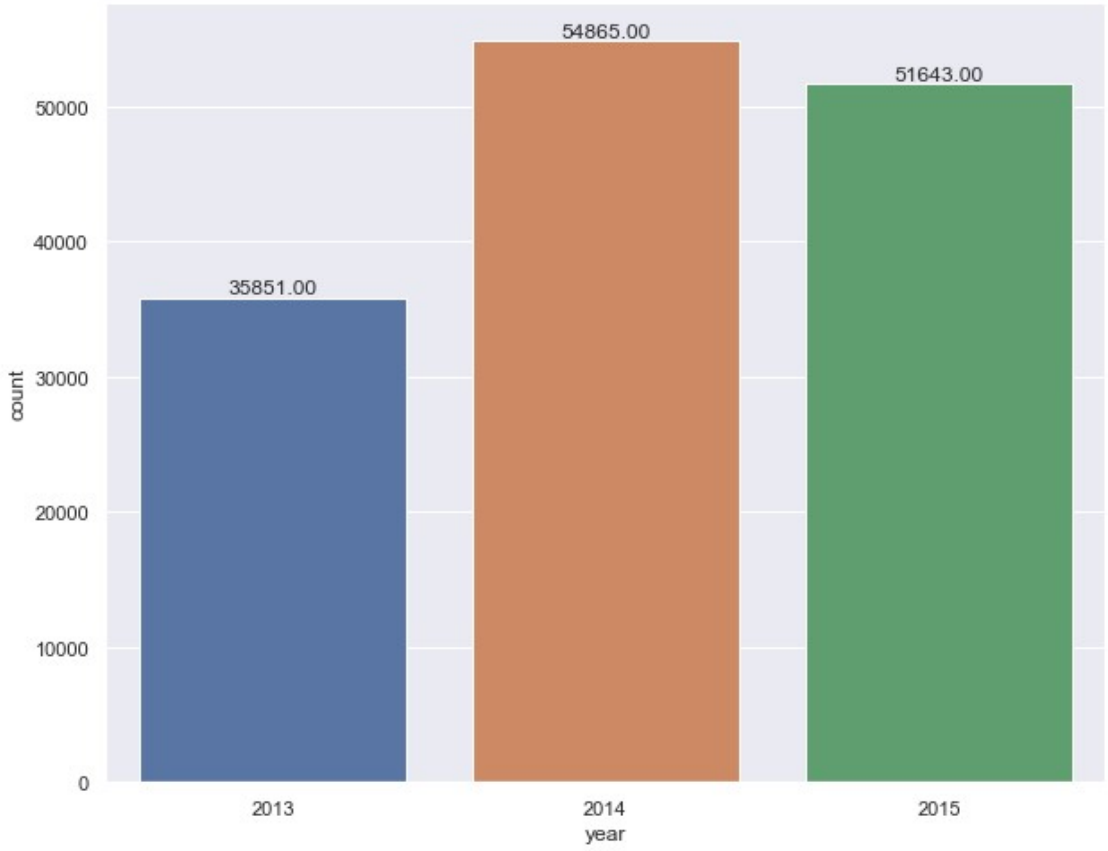


- Em todos os anos, Paid Marketing se sobressai em relação ao ticket médio da receita líquida quando comparado com o Referral.

A3. Número de clientes únicos por Ano e comparativo desse resultado 2013x2015 (em gráfico e tabela)

```
aux1 =
pd.DataFrame(df4[['customer_id','year']].value_counts()).reset_index()

aux1 = aux1.rename(columns={0:'qtd_buy'})
plt.figure(figsize=[10,8])
cx = sns.countplot(x='year',data=aux1)
for p in cx.patches:
    cx.annotate(format(p.get_height(), '.2f'),
                (p.get_x() + p.get_width() / 2, p.get_height()),
```

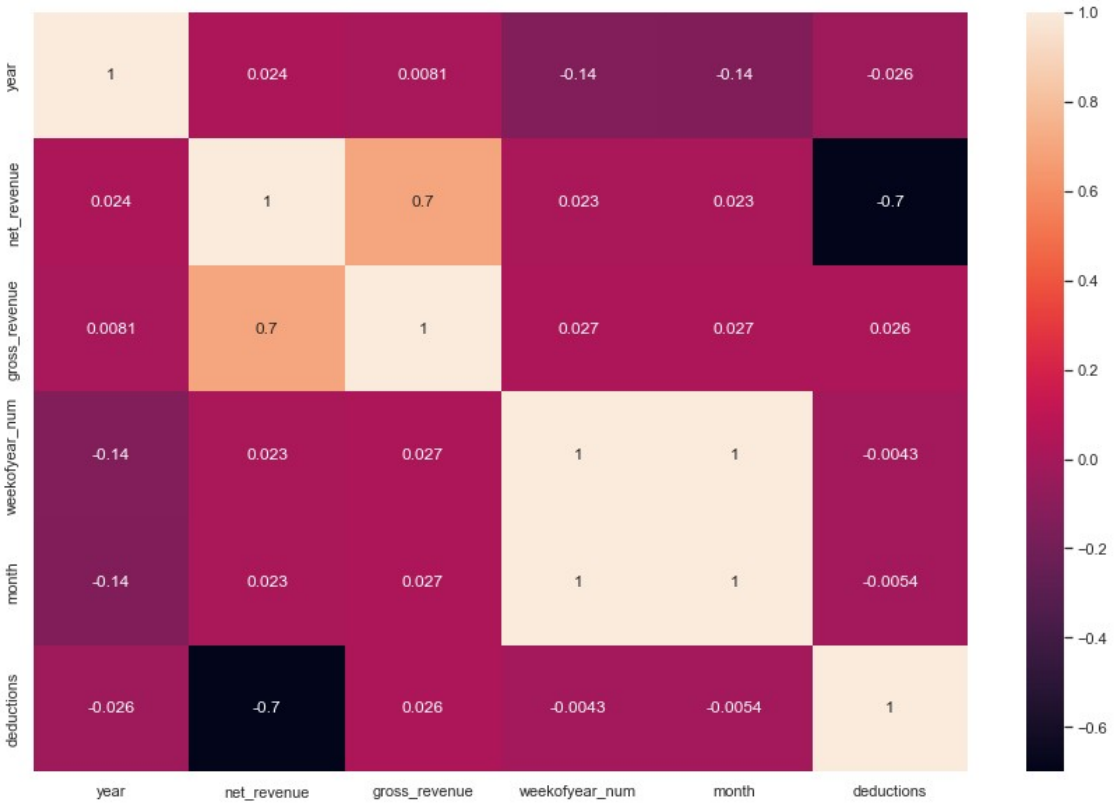
```
        ha = 'center',
        va = 'baseline',
        xytext = (0,2),
        textcoords = 'offset points')
```



4.3 Análise multivariada

```
corr =
df4.drop(columns=['boxes','customer_id']).corr(method='pearson')
sns.heatmap(corr,annot=True)
```

```
<AxesSubplot:>
```



Há uma boa correlação entre a receita líquida e a receita bruta (o que já era algo esperado). As semanas e os meses tem uma correlação fraca com a receita líquida/bruta. E por não termos mais variáveis para a produção do modelo, estaremos adotando apenas modelos de séries temporais

4.4 Análise para série temporal

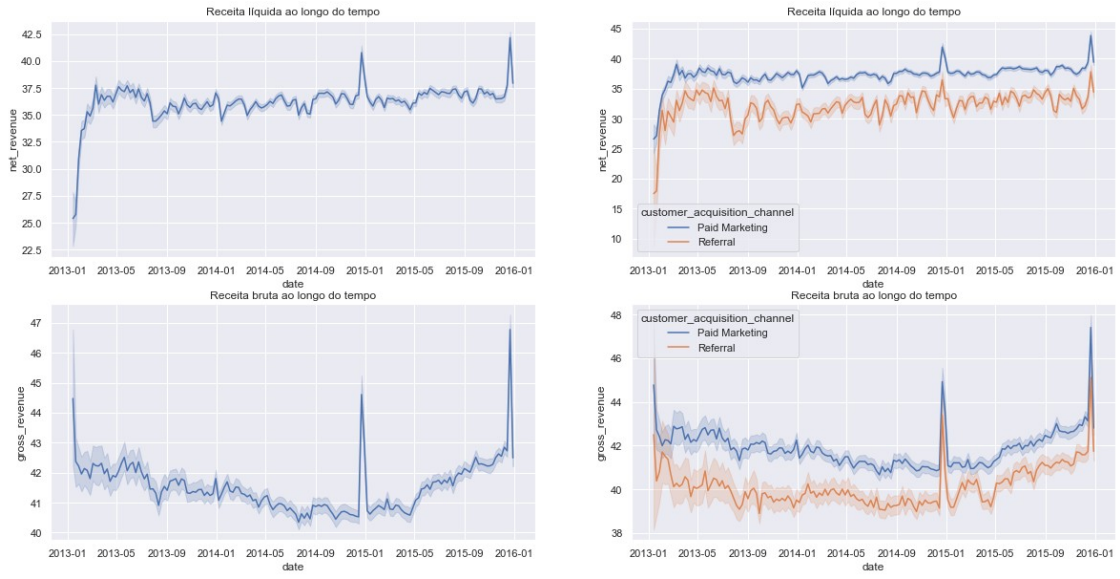
Nas séries temporais temos que observar dois componentes: Tendência (comportamento da curva) e ruídos (devido a sazonalidade ou ruído aleatório devido ao fenômeno).

4.4.1 Tendência ao longo do tempo

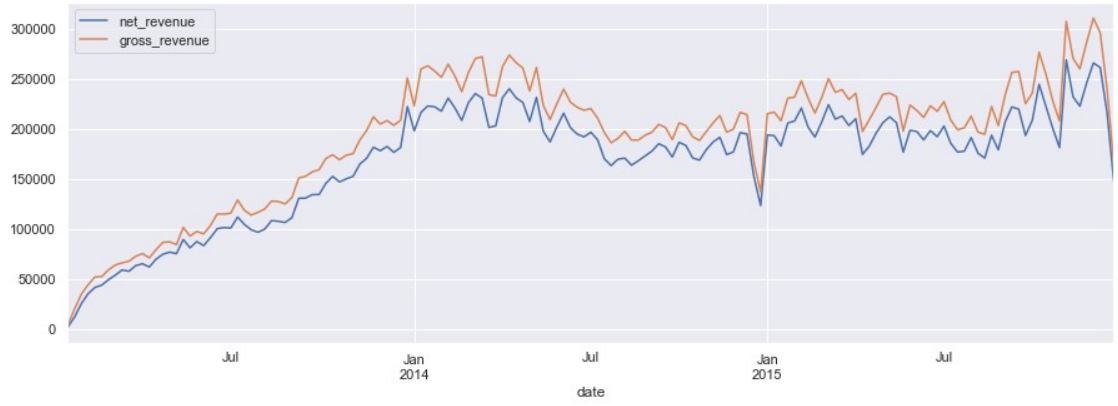
```
plt.figure(figsize=[20,10])
palette = sns.color_palette("Paired")
plt.subplot(2,2,1)
sns.lineplot(x='date',y='net_revenue',data=df4)
plt.title(label='Receita líquida ao longo do tempo')
plt.subplot(2,2,2)
sns.lineplot(x='date',y='net_revenue',data=df4,
hue='customer_acquisition_channel')
plt.title(label='Receita líquida ao longo do tempo')
plt.subplot(2,2,3)
sns.lineplot(x='date',y='gross_revenue',data=df4, palette=palette)
plt.title(label='Receita bruta ao longo do tempo')
plt.subplot(2,2,4)
sns.lineplot(x='date',y='gross_revenue',data=df4,
```

```
hue='customer_acquisition_channel')
plt.title(label='Receita bruta ao longo do tempo')
```

```
Text(0.5, 1.0, 'Receita bruta ao longo do tempo')
```

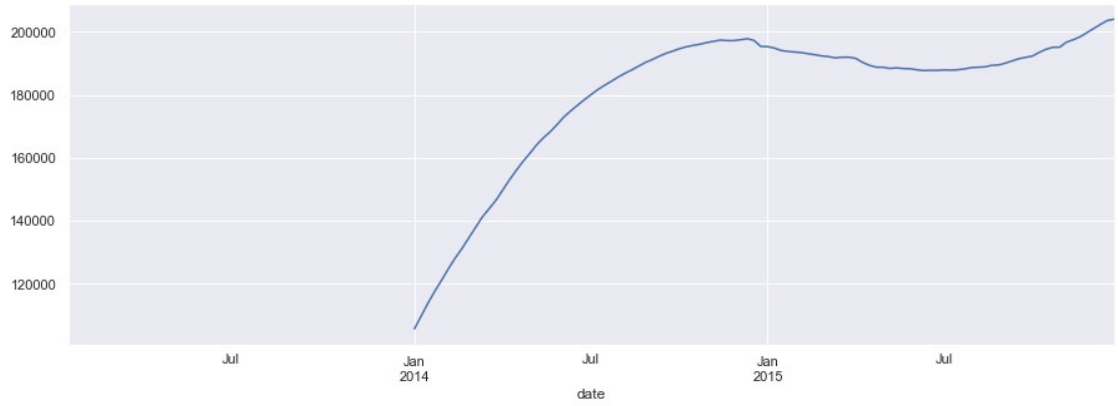



```
df_week=
df4[['net_revenue','gross_revenue','date']].groupby('date').sum()
df_week.plot(figsize=(15,5));
```



Uma das formas de se eliminar o efeito de ruídos pode ser também pela Média móvel simples. Ex:

```
# Analisando uma janela de 52 semanas que é equivalente a um ano
df_week.net_revenue.rolling(52).mean().plot(figsize=(15,5));
```



4.4.2 Sazonalidade

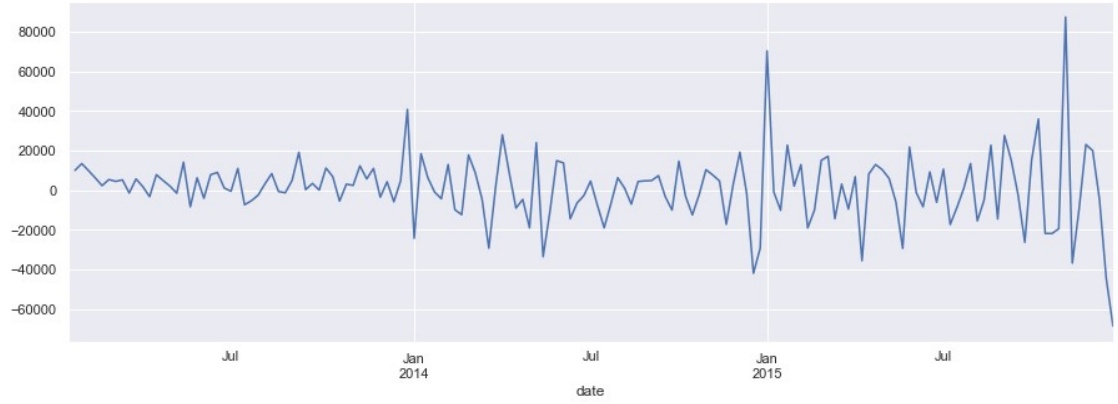
Nessa análise, precisamos eliminar o componente de tendência. Para isso, precisamos utilizar o conceito de diferenças.

Diferenças: O procedimento utilizado para eliminar a tendência é conhecido na literatura como procedimento para transformação de uma série não estacionária em uma série estacionária.

A transformação mais comum consiste em tomar diferenças sucessivas da série original até se obter uma série estacionária.

```
df_week.net_revenue.diff().plot(figsize=(15,5))
```

```
<AxesSubplot:xlabel='date'>
```



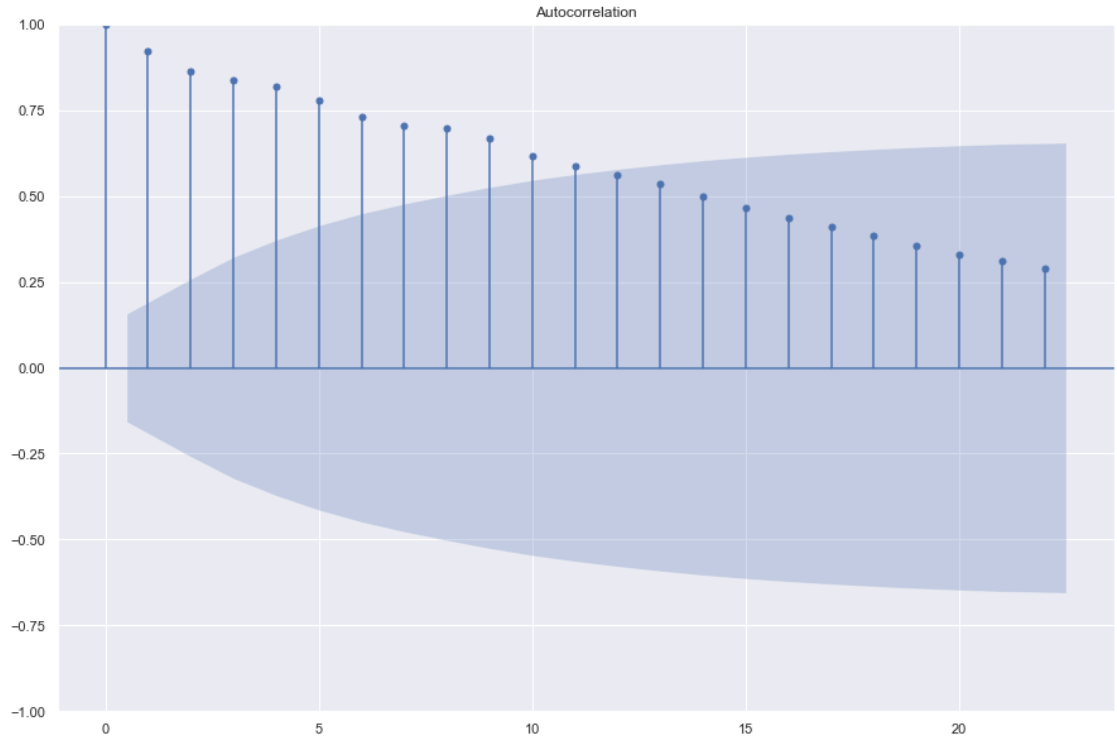
A partir do gráfico acima percebe-se que não há um comportamento sanzonal bem definido para uma janela semanal.

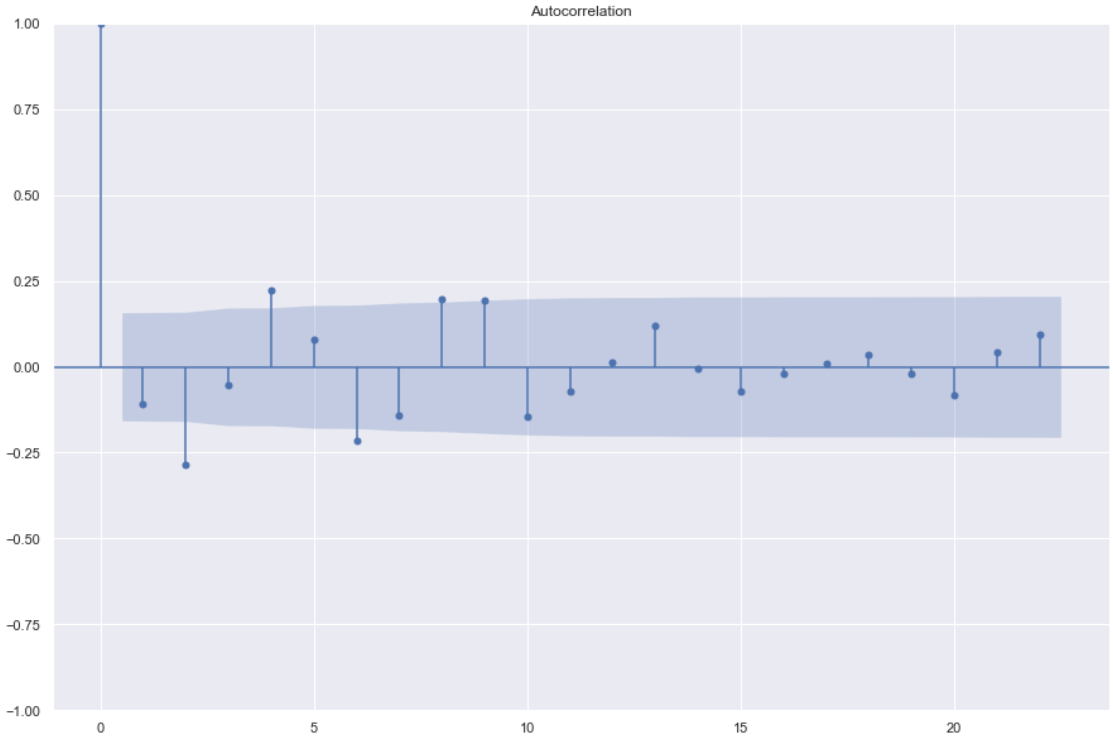
```
df_week.net_revenue.diff().groupby(df_week.index.month).mean().plot(ki
nd='bar',figsize=(15,5))
```

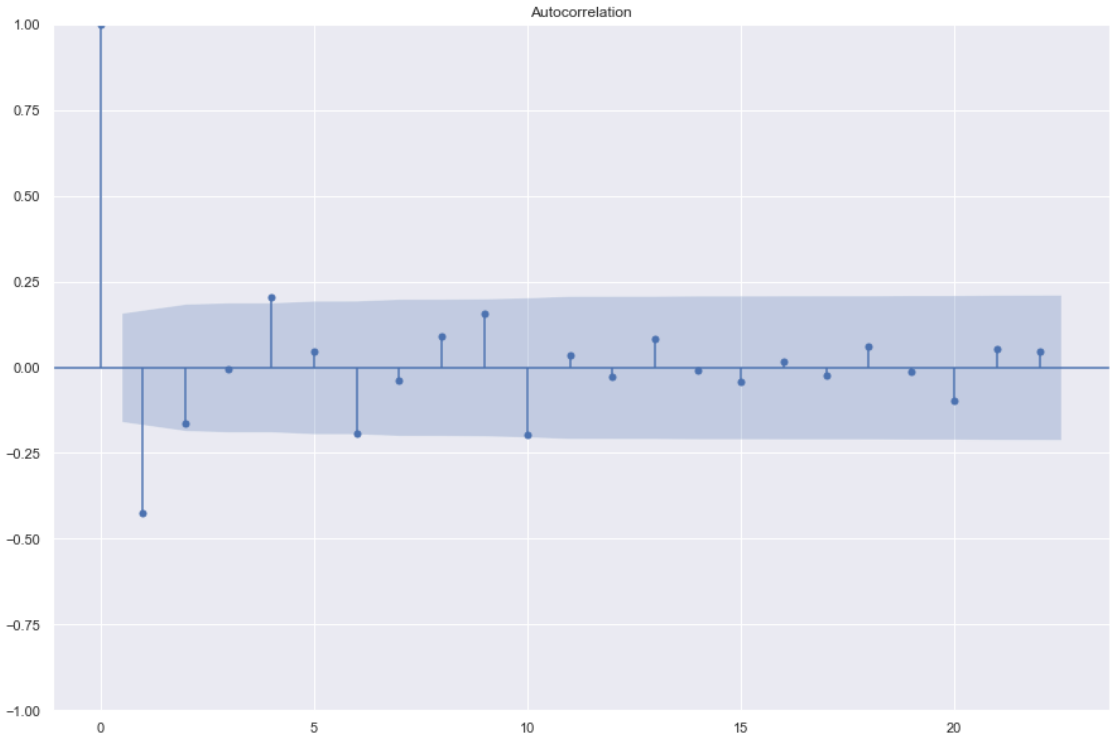
```
<AxesSubplot:xlabel='date'>
```


```
plt.figure(figsize=[15,3])
plot_acf(df_week.net_revenue)
plot_acf(df_week.net_revenue.diff().dropna())
plot_acf(df_week.net_revenue.diff().diff().dropna())
plt.show()
```

```
<Figure size 1080x216 with 0 Axes>
```




Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

5.0 Data Preparation

5.1 Transformation

```
df5 = df4.copy()
```

Exemplo:

- Normalização/ Padronização - Distribuição normal (Numérico)
- Rescaling - Presença de outliers (Numérico)
- Encoding: Label Encoder, One Hot Encoder, Ordinal e etc... (Categórica)
- Transformation (Variável numérica cíclica - Horas, meses, dia, semanas e etc)

```
# Label encoder
le = LabelEncoder()
df5['customer_acquisition_channel'] =
le.fit_transform(df5['customer_acquisition_channel'])
df5['boxes'] = le.fit_transform(df5['boxes'])

# Transformation Cycle
# 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)))

# week of year
df5['weekofyear_sin']= df5['weekofyear_num'].apply(lambda x: np.sin(x
* (2. * np.pi/52)))
df5['weekofyear_cos']= df5['weekofyear_num'].apply(lambda x: np.cos(x
* (2. * np.pi/52)))
```

6.0 Feature Selection

```
df6 = df5.copy()
```

Principais métodos:

- Seleção univariada (Filter Methods)
- Seleção por importância (Embedded Methods)
- Seleção por Subset (Wrapper Methods)

```
# Filtrando variáveis dependentes
df6 = df6.drop(columns=['week', 'customer_id', 'gross_revenue',
                        'weekofyear','deductions',
                        'weekofyear_num','month'],axis=1)
```

6.1 Split Dataframe into Training and test Dataset

Previsão de 6 meses

Dataset for ML

```
drop_cols = ['date','net_revenue']

df_train_origin = df6[df6['date'] < '2015-06-30']
df_test_origin = df6[df6['date'] > '2015-06-30']

# Train
x_train = df_train_origin.drop(columns=drop_cols, axis=1)
y_train = df_train_origin['net_revenue'].values

# Test
x_test = df_test_origin.drop(columns=drop_cols, axis=1)
y_test = df_test_origin['net_revenue'].values
```

Dataset for Forecasting

```
df_forecasting = df_week.copy() # 1º Diferenciação
df_forecasting = df_forecasting.reset_index()

df_forecasting_train = df_forecasting[df_forecasting['date'] < '2015-
06-30'].drop(columns='gross_revenue',axis=1).set_index('date')
df_forecasting_test = df_forecasting[df_forecasting['date'] > '2015-
06-30'].drop(columns='gross_revenue',axis=1).set_index('date')
```

7.0 Machine Learning Modeling

Utilizando a estratégia da 'Navalha de Ockaham', adotaremos inicialmente os modelos mais simples, Ou seja, mais fáceis de serem explicados.

7.1 Average Model

```
aux1 = df_test_origin.copy()

# Predictions
aux2 =
aux1[['date','net_revenue']].groupby('date').mean().reset_index().rena
me(columns={'net_revenue':'predictions'})
aux3 = pd.merge(aux1, aux2, how='left', on='date')
yhat_baseline = aux3['predictions']

# performance
baseline_result = ml_error('Average Model', y_test, yhat_baseline)
baseline_result
```

```
      Model Name       MAE  MAPE       RMSE
0  Average Model  8.123849   inf  13.471692
```

O MAPE deu INF por causa de algum valor no Y_test ser igual a zero. Afinal, quando um determinado valor é dividido por zero, temos aí uma indeterminação.

7.2 Linear regression model

```
# Model
lr = LinearRegression().fit(x_train,y_train)

# Prediction
yhat_lr = lr.predict(x_test)

# performance
lr_result = ml_error('Linear Regression', y_test,yhat_lr)
lr_result
```

```
          Model Name       MAE  MAPE       RMSE
0  Linear Regression  8.454716   inf  12.867058
```

O MAE e o RMSE foram bem menores que o baseline

- Os resultados foram muito promissores

7.3 Random Forest Regression

Esse é um tipo de modelo não-linear, geralmente é ultilizado em problemas mais complexos.

```
# Model
rf = RandomForestRegressor(n_estimators=1, n_jobs=-1,
random_state=42).fit(x_train,y_train)

# Prediction
yhat_rf = rf.predict(x_test)

# performance
rf_result = ml_error('Random Forest', y_test,yhat_rf)
rf_result
```

```
      Model Name       MAE  MAPE       RMSE
0  Random Forest  8.376739   inf  12.758662
```

7.4 ARIMA Model

AutoRegressive Integrated Moving Average (ARIMA) é um modelo de previsão de séries temporais que incorpora medidas de autocorrelação para modelar estruturas temporais dentro dos dados de séries temporais para prever valores futuros.

Modelos autorregressivos são conceitualmente semelhantes à regressão linear , as suposições feitas por este último também valem aqui. Os dados de séries temporais devem ser estacionários para remover qualquer correlação e colinearidade óbvias com os dados anteriores. Em dados de séries temporais estacionárias, as propriedades ou o valor de uma observação de amostra não dependem do registro de data e hora em que ela é observada.

Suponha que os valores passados em seus dados afetem os valores atuais ou futuros ou possam prever tendências futuras com base em flutuações recentes. Nesse caso, a previsão de séries temporais é a solução para esse problema de regressão. Vários outros modelos de previsão de séries temporais dependem da incorporação de mudanças sucessivas ou desenvolvimentos mais recentes nos dados para prever tendências futuras.

- **EM SUMA, Qualquer série temporal 'não sazonal' que exiba padrões e não seja um ruído branco aleatório pode ser modelada com modelos ARIMA.**

```
model = ARIMA(df_forecasting_train.values,order=(1,1,4))
model_fit = model.fit()

start = len(df_forecasting_train)
end = len(df_forecasting_train) + len(df_forecasting_test) - 1
predictions = model_fit.predict(start, end,
                                typ = 'levels')


# performance
arima_result = ml_error('ARIMA Model', df_forecasting_test.values,
predictions)
arima_result
```

```
    Model Name            MAE      MAPE          RMSE
0  ARIMA Model  30570.303726  0.131722  39452.189904
```

7.5 Compare Model's Performance

```
modelling_result =
pd.concat([baseline_result,lr_result,rf_result,arima_result])
modelling_result = modelling_result.sort_values('RMSE')

modelling_result
```

```
          Model Name            MAE      MAPE          RMSE
0      Random Forest       8.376739       inf     12.758662
0  Linear Regression       8.454716       inf     12.867058
0      Average Model       8.123849       inf     13.471692
0        ARIMA Model  30570.303726  0.131722  39452.189904
```

7.6 Cross Validation Time Series

```
def cross_validation(df_train_origin, kfold, model_name, model,
verbose=True):
    mae_list = []
    mape_list = []
    rmse_list = []
    for k in reversed(range(1,kfold+1)):
        if verbose:
            print('\nKfold number: {}'.format(k))
        # Start and end date for validation
        validation_start_date = df_train_origin['date'].max() -
```

```
datetime.timedelta(days = k*30*6) #
        validation_end_date = df_train_origin['date'].max() -
datetime.timedelta(days = (k-1)*30*6)

        # Filtering
        training = df_train_origin[df_train_origin['date']<
validation_start_date]
        validation = df_train_origin[(df_train_origin['date']>=
validation_start_date) & (df_train_origin['date'] <=
validation_end_date)]

        # Training and Validation dataset
        # Training
        xtraining = training.drop(['date','net_revenue'], axis=1)
        ytraining = training['net_revenue']

        # Validation
        xvalidation = validation.drop(['date','net_revenue'], axis=1)
        yvalidation = validation['net_revenue']

        # model
        m = model.fit(xtraining,ytraining)

        # Predict
        yhat = m.predict(xvalidation)

        # Performance
        m_result = ml_error(model_name, yvalidation, yhat)

        # Store performance
        mae_list.append(m_result['MAE'])
        mape_list.append(m_result['MAPE'])
        rmse_list.append(m_result['RMSE'])

        pd.DataFrame({'Model Name': model_name,
                      'MAE CV':
np.round(np.mean(mae_list),2).astype(str) + ' +/- ' +
np.round(np.std(mae_list),2).astype(str),
                      'MAPE CV':
np.round(np.mean(mape_list),2).astype(str) + ' +/- ' +
np.round(np.std(mape_list),2).astype(str),
                      'RMSE CV':
np.round(np.mean(rmse_list),2).astype(str) + ' +/- ' +
np.round(np.std(rmse_list),2).astype(str)})


xtraining = df_train_origin.drop(['date','net_revenue'], axis=1)

#model = LinearRegression()
#a = cross_validation(xtraining, 5, 'Linear Regression', model)
```