Data Science - Regressão Linear II

1.2 Conhecendo o Dataset

Importando a biblioteca pandas

https://pandas.pydata.org/

In [10]:

```
import pandas as pd
```

O Dataset e o Projeto

Descrição:

O mercado imobiliário vem sendo objeto de diversos estudos e pesquisas nos últimos tempos. A crise financeira que afeta a economia tem afetado significativamente os investimentos e ganhos advindos deste setor. Este cenário incentiva o aumento do interesse por estudos de previsão de demanda baseados em características deste mercado, dos imóveis e do entorno destes imóveis.

Neste contexto o objetivo principal do nosso projeto é desenvolver um sistema de avaliação imobiliária utilizando a metodologia de regressões lineares que é uma das técnicas de machine learning.

Nosso *dataset* é uma amostra aleatória de tamanho 5000 de imóveis disponíveis para venda no município do Rio de Janeiro.

Dados:

- **Valor** - Valor (R$) de oferta do imóvel
- **Area** - Área do imóvel em m^2
- **Dist_Praia** - Distância do imóvel até a praia (km) (em linha reta)
- **Dist_Farmacia** - Distância do imóvel até a farmácia mais próxima (km) (em linha reta)

Leitura dos dados

In [11]:

```
dados = pd.read_csv('../Dados/dataset.csv', sep=';')
```

Visualizar os dados

In [12]:

```
dados.head()
```

Out[12]:

	Valor	Area	Dist_Praia	Dist_Farmacia
0	4600000	280	0.240925	0.793637
1	900000	208	0.904136	0.134494
2	2550000	170	0.059525	0.423318
3	550000	100	2.883181	0.525064
4	2200000	164	0.239758	0.192374

Verificando o tamanho do dataset

In [13]:

```
dados.shape
```

Out[13]:

```
(5000, 4)
```

1.3 Análises Preliminares

Estatísticas descritivas

In [14]:

```
dados.describe().round(2)
```

Out[14]:

	Valor	Area	Dist_Praia	Dist_Farmacia
count	5000.00	5000.00	5000.00	5000.00
mean	1402926.39	121.94	3.02	0.50
std	1883268.85	90.54	3.17	0.29
min	75000.00	16.00	0.00	0.00
25%	460000.00	70.00	0.44	0.24
50%	820000.00	93.00	1.48	0.50
75%	1590000.00	146.00	5.61	0.75
max	25000000.00	2000.00	17.96	1.00

Matriz de correlação

> O **coeficiente de correlação** é uma medida de associação linear entre duas variáveis e situa-se entre **-1** e **+1** sendo que **-1** indica associação negativa perfeita e **+1** indica associação positiva perfeita.

In [15]:

```
dados.corr().round(4)
```

Out[15]:

	Valor	Area	Dist_Praia	Dist_Farmacia
Valor	1.0000	0.7110	-0.3665	-0.0244
Area	0.7110	1.0000	-0.2834	-0.0310
Dist_Praia	-0.3665	-0.2834	1.0000	0.0256
Dist_Farmacia	-0.0244	-0.0310	0.0256	1.0000

2.1 Comportamento da Variável Dependente (Y)

Importando biblioteca seaborn

https://seaborn.pydata.org/

> O Seaborn é uma biblioteca Python de visualização de dados baseada no matplotlib. Ela fornece uma interface de alto nível para desenhar gráficos estatísticos.

In [16]:
```
import seaborn as sns
```

Configurações de formatação dos gráficos

In [17]:
```
# palette -> Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn,
BuGn_r, BuPu, BuPu_r, CMRmap, CMRmap_r, Dark2, Dark2_r, GnBu, GnBu_r,
Greens, Greens_r, Greys, Greys_r, OrRd, OrRd_r, Oranges, Oranges_r,
PRGn, PRGn_r, Paired, Paired_r, Pastel1, Pastel1_r, Pastel2,
Pastel2_r, PiYG, PiYG_r, PuBu, PuBuGn, PuBuGn_r, PuBu_r, PuOr, PuOr_r,
PuRd, PuRd_r, Purples, Purples_r, RdBu, RdBu_r, RdGy, RdGy_r, RdPu,
RdPu_r, RdYlBu, RdYlBu_r, RdYlGn, RdYlGn_r, Reds, Reds_r, Set1,
Set1_r, Set2, Set2_r, Set3, Set3_r, Spectral, Spectral_r, Wistia,
Wistia_r, YlGn, YlGnBu, YlGnBu_r, YlGn_r, YlOrBr, YlOrBr_r, YlOrRd,
YlOrRd_r, afmhot, afmhot_r, autumn, autumn_r, binary, binary_r, bone,
bone_r, brg, brg_r, bwr, bwr_r, cividis, cividis_r, cool, cool_r,
coolwarm, coolwarm_r, copper, copper_r, cubehelix, cubehelix_r, flag,
flag_r, gist_earth, gist_earth_r, gist_gray, gist_gray_r, gist_heat,
gist_heat_r, gist_ncar, gist_ncar_r, gist_rainbow, gist_rainbow_r,
gist_stern, gist_stern_r, gist_yarg, gist_yarg_r, gnuplot, gnuplot2,
gnuplot2_r, gnuplot_r, gray, gray_r, hot, hot_r, hsv, hsv_r, icefire,
icefire_r, inferno, inferno_r, jet, jet_r, magma, magma_r, mako,
mako_r, nipy_spectral, nipy_spectral_r, ocean, ocean_r, pink, pink_r,
plasma, plasma_r, prism, prism_r, rainbow, rainbow_r, rocket,
rocket_r, seismic, seismic_r, spring, spring_r, summer, summer_r,
tab10, tab10_r, tab20, tab20_r, tab20b, tab20b_r, tab20c, tab20c_r,
terrain, terrain_r, viridis, viridis_r, vlag, vlag_r, winter, winter_r
sns.set_palette("Accent")

# style -> white, dark, whitegrid, darkgrid, ticks
sns.set_style("darkgrid")
```

Box plot da variável *dependente* (y)

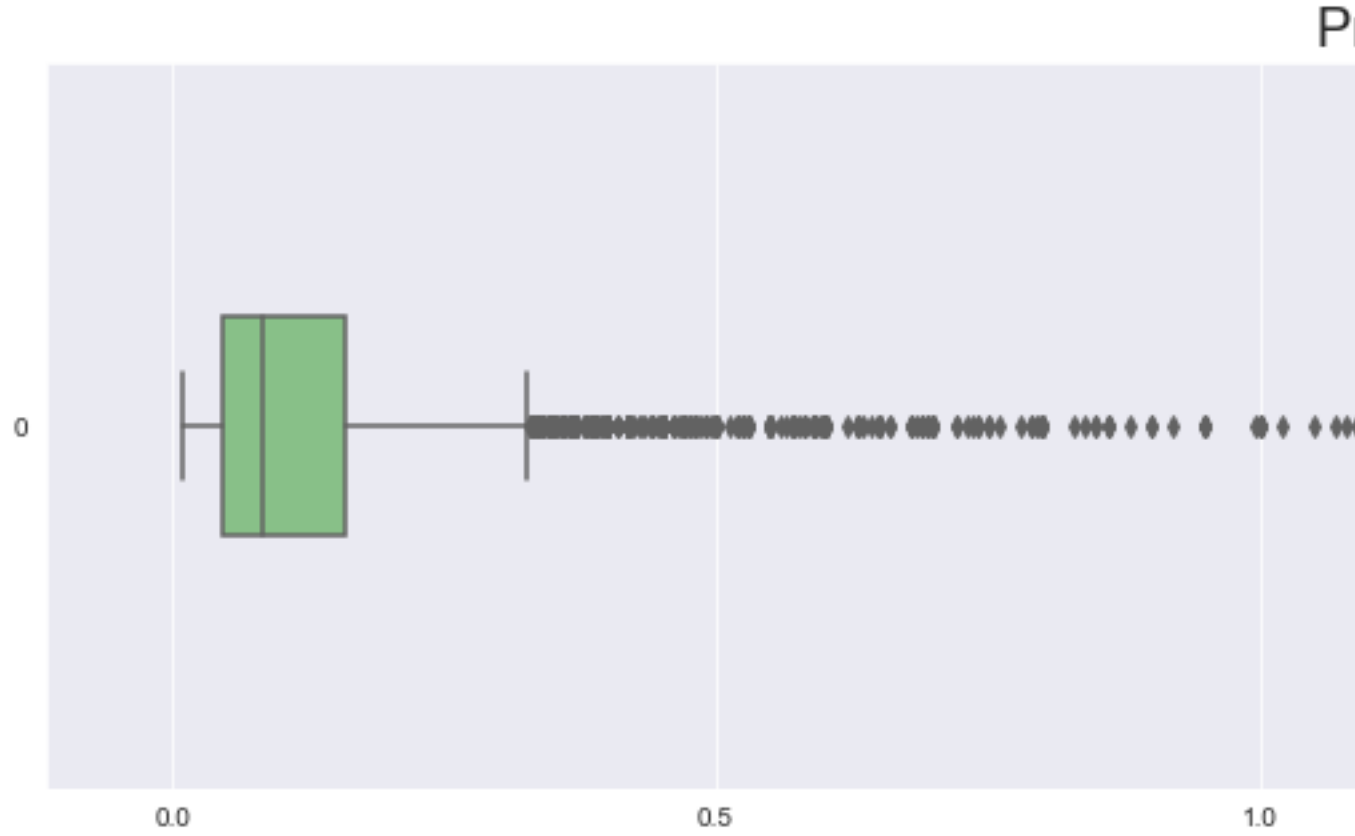
https://seaborn.pydata.org/generated/seaborn.boxplot.html?highlight=boxplot#seaborn.boxplot

In [18]:

```
ax = sns.boxplot(data=dados['Valor'], orient='h', width=0.3)
ax.figure.set_size_inches(20, 5)
ax.set_title('Preço dos Imóveis', fontsize=20)
ax.set_xlabel('Reais', fontsize=16)
ax
```

Out[18]:

```
<AxesSubplot:title={'center':'Preço dos Imóveis'}, xlabel='Reais'>
```



2.2 Distribuição de Frequências

Distribuição de frequências da variável *dependente* (y)

https://seaborn.pydata.org/generated/seaborn.distplot.html?highlight=distplot#seaborn.distplot

In [19]:

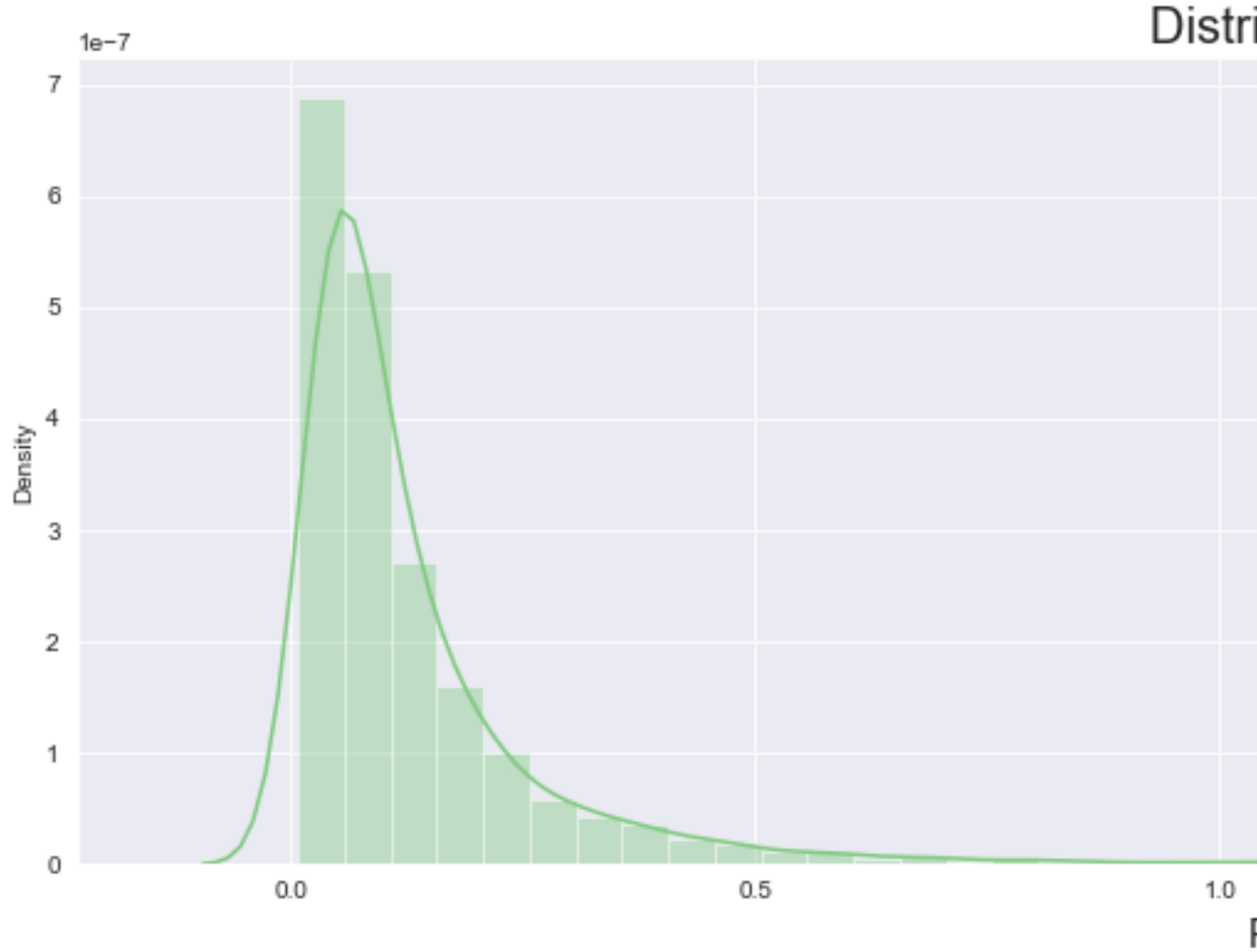
```
ax = sns.distplot(dados['Valor'])
ax.figure.set_size_inches(20, 6)
```

```
ax.set_title('Distribuição de Frequências', fontsize=20)
ax.set_xlabel('Preço dos Imóveis (R$)', fontsize=16)
ax
```

```
C:\Users\EDWARD SOUSA\anaconda3\lib\site-
packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[19]:

```
<AxesSubplot:title={'center':'Distribuição de Frequências'},
xlabel='Preço dos Imóveis (R$)', ylabel='Density'>
```



2.3 Dispersão Entre as Variáveis

Gráficos de dispersão entre as variáveis do dataset

seaborn.pairplot

https://seaborn.pydata.org/generated/seaborn.pairplot.html?highlight=pairplot#seaborn.pairplot

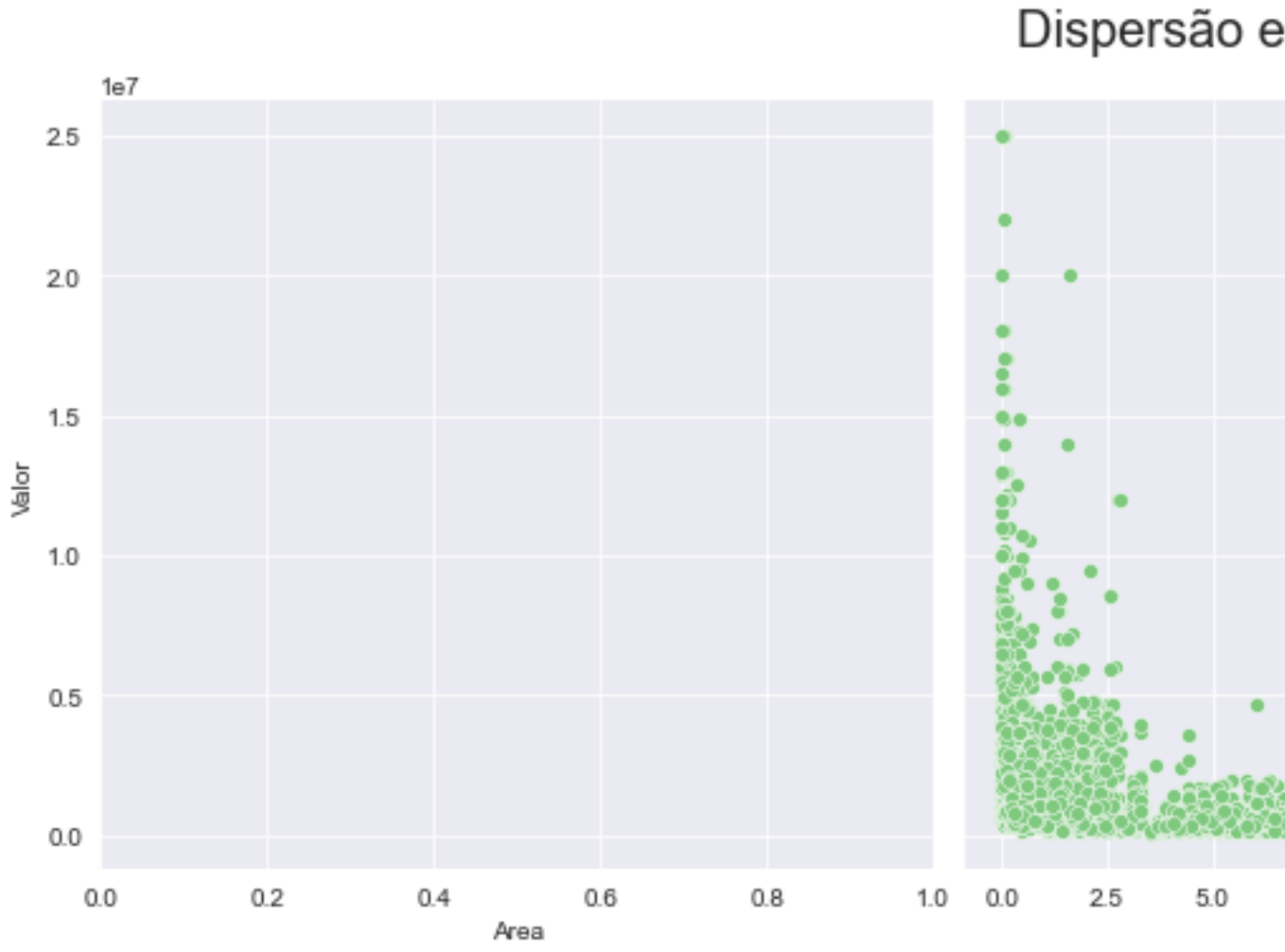
Plota o relacionamento entre pares de variáveis em um dataset.

In [20]:

```
ax = sns.pairplot(dados, y_vars='Valor', x_vars=['Area', 'Dist_Praia',
'Dist_Farmacia'], height=5)
ax.fig.suptitle('Dispersão entre as Variáveis', fontsize=20, y=1.05)
ax
```

Out[20]:

```
<seaborn.axisgrid.PairGrid at 0x229218c71f0>
```



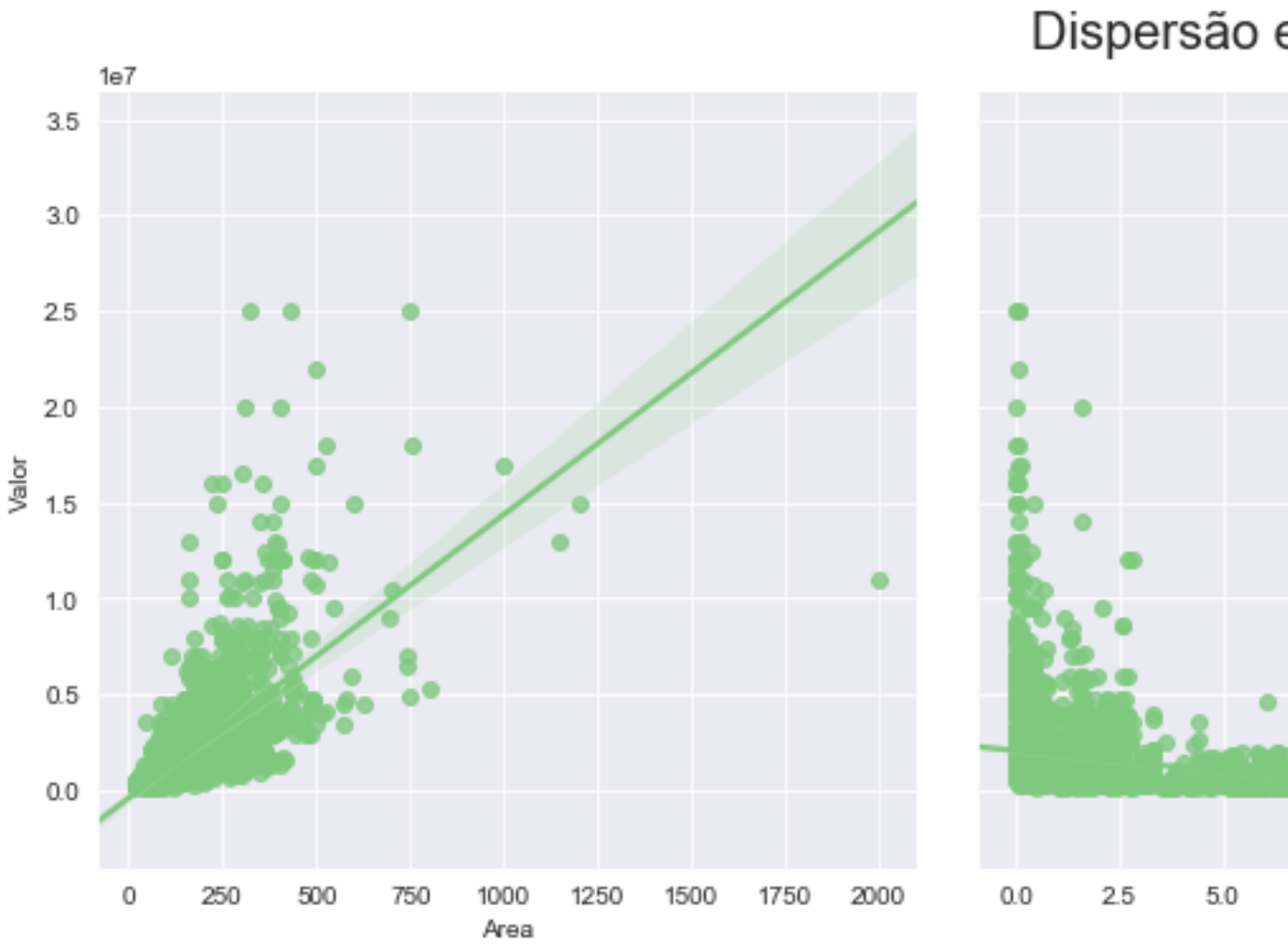
In [55]:

```
ax = sns.pairplot(dados, y_vars='Valor', x_vars=['Area', 'Dist_Praia',
'Dist_Farmacia'], kind='reg', height=5)
```

```
ax.fig.suptitle('Dispersão entre as Variáveis', fontsize=20, y=1.05)
ax
C:\Users\Alura\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713:
FutureWarning: Using a non-tuple sequence for multidimensional
indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`.
In the future this will be interpreted as an array index,
`arr[np.array(seq)]`, which will result either in an error or a
different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[55]:

```
<seaborn.axisgrid.PairGrid at 0x18b937284e0>
```



3.1 Transformando os Dados

Distribuição Normal

Por quê?

> Testes paramétricos assumem que os dados amostrais foram coletados de uma população com distribuição de probabilidade conhecida. Boa parte dos testes estatísticos assumem que os dados seguem uma distribuição normal (t de Student, intervalos de confiança etc.).

Importando biblioteca numpy

In [21]:

```
import numpy as np
```

Aplicando a transformação logarítmica aos dados do *dataset*

https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.log.html

In [22]:

```
np.log(1)
```

Out[22]:

```
0.0
```

In [24]:

```
dados['log_Valor'] = np.log(dados['Valor'])
dados['log_Area'] = np.log(dados['Area'])
dados['log_Dist_Praia'] = np.log(dados['Dist_Praia'] + 1)
dados['log_Dist_Farmacia'] = np.log(dados['Dist_Farmacia'] + 1)
```

In [26]:

```
dados.head()
```

Out[26]:

	Valor	Area	Dist_Praia	Dist_Farmacia	log_Valor	log_Area	log_Dist_Praia	log_Dist_Farmacia
0	4600000	280	0.240925	0.793637	15.341567	5.634790	0.215857	0.584245
1	900000	208	0.904136	0.134494	13.710150	5.337538	0.644028	0.126187
2	2550000	170	0.059525	0.423318	14.751604	5.135798	0.057821	0.352991
3	550000	100	2.883181	0.525064	13.217674	4.605170	1.356655	0.422036
4	2200000	164	0.239758	0.192374	14.603968	5.099866	0.214916	0.175946

In [27]:

```
dados.Dist_Praia
```

Out[27]:

```
0       0.240925
1       0.904136
2       0.059525
3       2.883181
4       0.239758
```

```
         ...
4995    0.479357
4996    8.594487
4997    0.253138
4998    8.945226
4999    0.774444
Name: Dist_Praia, Length: 5000, dtype: float64
```

Distribuição de frequências da variável *dependente transformada* (y)

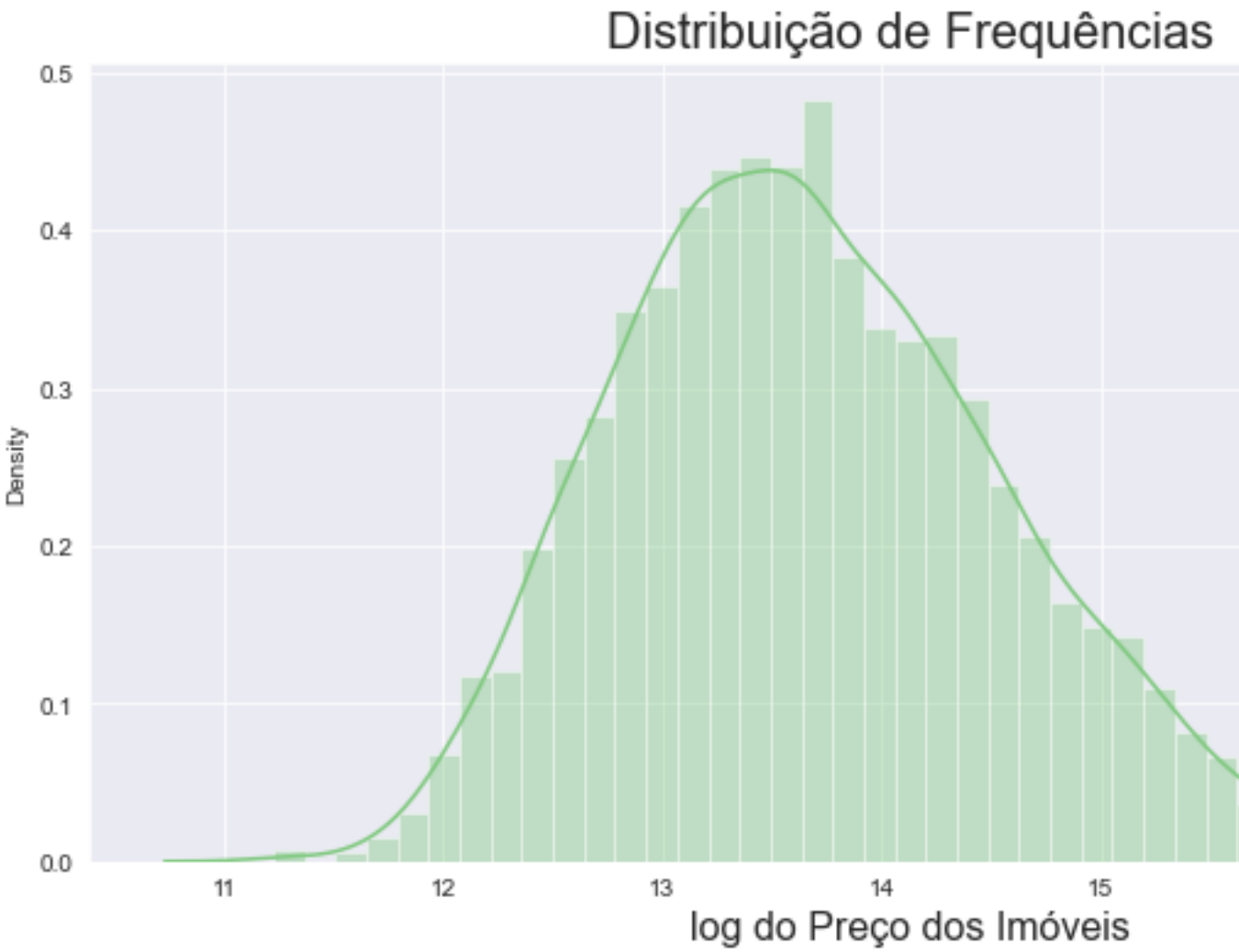
In [28]:

```
ax = sns.distplot(dados['log_Valor'])
ax.figure.set_size_inches(12, 6)
ax.set_title('Distribuição de Frequências', fontsize=20)
ax.set_xlabel('log do Preço dos Imóveis', fontsize=16)
ax
```

```
C:\Users\EDWARD SOUSA\anaconda3\lib\site-
packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[28]:

```
<AxesSubplot:title={'center':'Distribuição de Frequências'},
xlabel='log do Preço dos Imóveis', ylabel='Density'>
```



3.2 Verificando Relação Linear

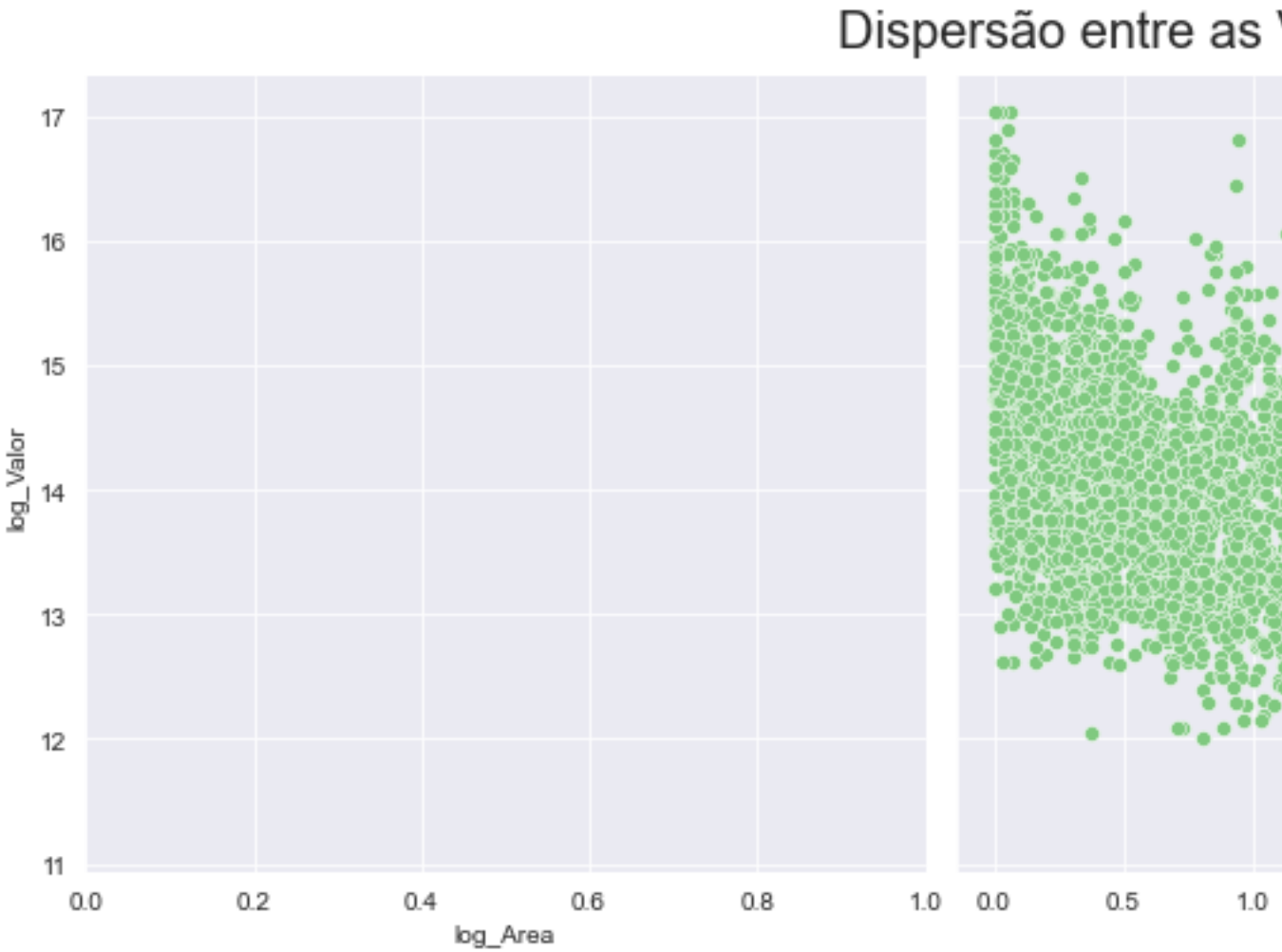
Gráficos de dispersão entre as variáveis transformadas do dataset

In [29]:

```
ax = sns.pairplot(dados, y_vars='log_Valor', x_vars=['log_Area',
'log_Dist_Praia', 'log_Dist_Farmacia'], height=5)
ax.fig.suptitle('Dispersão entre as Variáveis Transformadas',
fontsize=20, y=1.05)
ax
```

Out[29]:

```
<seaborn.axisgrid.PairGrid at 0x2292203ddf0>
```



4.1 Criando os *Datasets* de Treino e Teste

Importando o *train_test_split* da biblioteca *scikit-learn*

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

In [30]:

```
from sklearn.model_selection import train_test_split
```

Criando uma Series (pandas) para armazenar o Preço dos Imóveis (y)

In [32]:

```
y = dados['log_Valor']
```

Criando um DataFrame (pandas) para armazenar as variáveis explicativas (X)

In [33]:
```
X = dados[['log_Area', 'log_Dist_Praia', 'log_Dist_Farmacia']]
```

Criando os datasets de treino e de teste

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=2811)
```

Regressão Linear

> A análise de regressão diz respeito ao estudo da dependência de uma variável (a variável **dependente**) em relação a uma ou mais variáveis, as variáveis explanatórias, visando estimar e/ou prever o valor médio da primeira em termos dos valores conhecidos ou fixados das segundas.

scikit-learn (https://scikit-learn.org/stable/)

> O *scikit-learn* é um módulo Python especializado em soluções para *machine learning*.

Importando a biblioteca statsmodels

https://www.statsmodels.org/stable/index.html

In [37]:

```
import statsmodels.api as sm
```

Estimando o modelo com statsmodels

In [38]:

```
X_train_com_constante = sm.add_constant(X_train)
```

In [40]:

```
X_train_com_constante
```

Out[40]:

	const	log_Area	log_Dist_Praia	log_Dist_Farmacia
2661	1.0	5.945421	0.000000	0.382273
912	1.0	3.135494	0.972865	0.605015
3042	1.0	4.317488	1.794961	0.486594
141	1.0	3.401197	0.310455	0.599609
3854	1.0	5.676754	0.032193	0.101903
...	...	...	...	...
3657	1.0	5.075174	2.023480	0.333605
979	1.0	4.174387	2.296141	0.156465
2389	1.0	4.394449	1.367741	0.409727
447	1.0	3.951244	2.166841	0.217381
2008	1.0	4.406719	1.692269	0.537831

4000 rows × 4 columns

In [42]:

```
modelo_statsmodels = sm.OLS(y_train, X_train_com_constante, hasconst =
True).fit()
```

4.2 Avaliando o Modelo Estimado

Avaliando as estatísticas de teste do modelo

In [43]:

```
print(modelo_statsmodels.summary())
                            OLS Regression Results
==============================================================================
========
Dep. Variable:              log_Valor   R-squared:
0.805
Model:                            OLS   Adj. R-squared:
0.805
Method:                 Least Squares   F-statistic:
5495.
Date:                Fri, 09 Apr 2021   Prob (F-statistic):
0.00
```

```
Time:                        14:09:57   Log-Likelihood:
-2044.9
No. Observations:                4000   AIC:
4098.
Df Residuals:                    3996   BIC:
4123.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          9.3417      0.060    154.734      0.000       9.223       9.460
log_Area       1.0580      0.012     89.320      0.000       1.035       1.081
log_Dist_Praia -0.4905      0.009    -56.690      0.000      -0.508      -0.474
log_Dist_Farmacia -0.0167      0.032     -0.521      0.603      -0.080       0.046
==============================================================================
Omnibus:                       64.751   Durbin-Watson:                   1.971
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              106.858
Skew:                           0.136   Prob(JB):                     6.25e-24
Kurtosis:                       3.753   Cond. No.                         47.6
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [45]:

```
modelo_statsmodels.summary()
```

Out[45]:

OLS Regression Results

Dep. Variable:	log_Valor	**R-squared:**	0.805
Model:	OLS	**Adj. R-squared:**	0.805
Method:	Least Squares	**F-statistic:**	5495.
Date:	Fri, 09 Apr 2021	**Prob (F-statistic):**	0.00
Time:	14:11:17	**Log-Likelihood:**	-2044.9
No. Observations:	4000	**AIC:**	4098.
Df Residuals:	3996	**BIC:**	4123.

Df Model:	3					
Covariance Type:	nonrobust					
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	9.3417	0.060	154.734	0.000	9.223	9.460
log_Area	1.0580	0.012	89.320	0.000	1.035	1.081
log_Dist_Praia	-0.4905	0.009	-56.690	0.000	-0.508	-0.474
log_Dist_Farmacia	-0.0167	0.032	-0.521	0.603	-0.080	0.046

Omnibus:	64.751	**Durbin-Watson:**	1.971
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	106.858
Skew:	0.136	**Prob(JB):**	6.25e-24
Kurtosis:	3.753	**Cond. No.**	47.6

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

4.3 Modificando o Modelo e Avaliando Novamente o Ajuste

Criando um novo conjunto de variáveis explicativas (X)

In [46]:
```
X = dados[['log_Area', 'log_Dist_Praia']]
```

Criando os datasets de treino e de teste

In [47]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=2811)
```

Estimando o modelo com o statsmodels

In [48]:
```
X_train_com_constante = sm.add_constant(X_train)
```

In [49]:
```
modelo_statsmodels = sm.OLS(y_train, X_train_com_constante, hasconst =
True).fit()
```

Avaliando as estatísticas de teste do novo modelo

Teste de significância conjunta dos parâmetros

Prob (F-statistic) <= 0.05 (**OK**)

Teste de significância individual dos parâmetros

P>|t| <= 0.05 (**OK**)

In [50]:

```
print(modelo_statsmodels.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_Valor   R-squared:
0.805
Model:                            OLS   Adj. R-squared:
0.805
Method:                 Least Squares   F-statistic:
8244.
Date:                Fri, 09 Apr 2021   Prob (F-statistic):
0.00
Time:                        14:17:54   Log-Likelihood:
-2045.1
No. Observations:                4000   AIC:
4096.
Df Residuals:                    3997   BIC:
4115.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
const          9.3349      0.059    158.353      0.000       9.219
9.450
log_Area       1.0581      0.012     89.345      0.000       1.035
1.081
log_Dist_Praia    -0.4906      0.009    -56.709      0.000      -0.508
-0.474
==============================================================================
Omnibus:                       65.115   Durbin-Watson:
1.972
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
107.712
Skew:                           0.136   Prob(JB):
4.08e-24
```

```
Kurtosis:                       3.757   Cond. No.
46.1
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

5.1 Estimando o Modelo com os Dados de Treino

Importando *LinearRegression* e *metrics* da biblioteca *scikit-learn*

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

https://scikit-learn.org/stable/modules/classes.html#regression-metrics

In [51]:

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

Instanciando a classe *LinearRegression()*

In [52]:

```
modelo = LinearRegression()
```

Utilizando o método *fit()* do objeto "modelo" para estimar nosso modelo linear utilizando os dados de TREINO (y_train e X_train)

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.fit

In [53]:

```
modelo.fit(X_train, y_train)
```

Out[53]:

```
LinearRegression()
```

Obtendo o coeficiente de determinação (R^2) do modelo estimado com os dados de TREINO

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.score

Coeficiente de Determinação - R^2

O coeficiente de determinação (R^2) é uma medida resumida que diz quanto a linha de regressão ajusta-se aos dados. É um valor entra 0 e 1.

$$R^2(y, \hat{y}) = 1 - \frac {\sum_{i=0}^{n-1}(y_i-\hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i-\bar{y}_i)^2}$$

In [54]:

```
print('R² = {}'.format(modelo.score(X_train, y_train).round(3)))
```

```
R² = 0.805
```

Gerando previsões para os dados de TESTE (X_test) utilizando o método *predict()* do objeto "modelo"

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression.predict

In [55]:

```
y_previsto = modelo.predict(X_test)
```

Obtendo o coeficiente de determinação (R^2) para as previsões do nosso modelo

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html#sklearn.metrics.r2_score

In [56]:

```
print('R² = %s' % metrics.r2_score(y_test, y_previsto).round(3))
```

```
R² = 0.79
```

5.2 Obtendo Previsões Pontuais

Dados de entrada

In [57]:

```
entrada = X_test[0:1]
entrada
```

Out[57]:

	log_Area	log_Dist_Praia
1006	5.273	1.282769

Gerando previsão pontual

In [59]:
```
modelo.predict(entrada)[0]
```

Out[59]:
```
14.28482006184788
```

Invertendo a transformação para obter a estimativa em R$

https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.exp.html

In [60]:
```
np.exp(modelo.predict(entrada)[0])
```

Out[60]:
```
1598889.784779439
```

Criando um simulador simples

In [61]:
```
Area = 150
Dist_Praia = 1
entrada = [[np.log(Area), np.log(Dist_Praia + 1)]]

print('R$ {0:.2f}'.format(np.exp(modelo.predict(entrada)[0])))
R$ 1617664.12
```

5.3 Interpretação dos Coeficientes Estimados

Obtendo o intercepto do modelo

> O **intercepto** representa o efeito médio em Y (Preço do Imóveis) tendo todas as variáveis explicativas excluídas do modelo. No caso do modelo log-linear este coeficiente deve ser transformado com o uso da função exponencial para ser apresentado em R$.

In [62]:
```
modelo.intercept_
```
Out[62]:
```
9.334916409800323
```
In [63]:
```
np.exp(modelo.intercept_)
```
Out[63]:
```
11326.681428069782
```

Obtendo os coeficientes de regressão

Os **coeficientes de regressão** β_2 e β_3 são conhecidos como **coeficientes parciais de regressão** ou **coeficientes parciais angulares**.

Um aspecto interessante do modelo log-linear, que o tornou muito utilizado nos trabalhos aplicados, é que os coeficientes angulares β_2 e β_3 medem as elasticidades de Y em relação a X_2 e X_3, isto é, a variação percentual de Y correspondente a uma dada variação percentual (pequena) em X_2 e X_3.

In [64]:
```
modelo.coef_
```
Out[64]:
```
array([ 1.05807818, -0.49061226])
```

Confirmando a ordem das variáveis explicativas no DataFrame

In [65]:
```
X.columns
```
Out[65]:
```
Index(['log_Area', 'log_Dist_Praia'], dtype='object')
```

Criando uma lista com os nomes das variáveis do modelo

In [66]:
```
index = ['Intercepto', 'log Área', 'log Distância até a Praia']
```

Criando um DataFrame para armazenar os coeficientes do modelo

https://docs.scipy.org/doc/numpy/reference/generated/numpy.append.html?#numpy.append

In [67]:
```
pd.DataFrame(data=np.append(modelo.intercept_, modelo.coef_),
index=index, columns=['Parâmetros'])
```

Out[67]:

	Parâmetros
Intercepto	9.334916
log Área	1.058078
log Distância até a Praia	-0.490612

Interpretação dos Coeficientes Estimados

Intercepto → Excluindo o efeito das variáveis explicativas ($X_2=X_3=0$) o efeito médio no Preço dos Imóveis seria de **R$ 11.326,68** (exp[9.334916]).

Área (m^2) → Mantendo-se o valor de X_3 (Distância até a Praia) constante, um acréscimo de 1% na Área de um imóvel gera, em média, um acréscimo de **1.06%** no Preço do Imóvel.

Distância até a Praia (km) → Mantendo-se o valor de X_2 (Área) constante, um acréscimo de 1% na

Distância de um imóvel até a praia gera, em média, um decréscimo de **0.49%** no Preço do Imóvel.

5.4 Análises Gráficas dos Resultados do Modelo

Gerando as previsões do modelo para os dados de TREINO

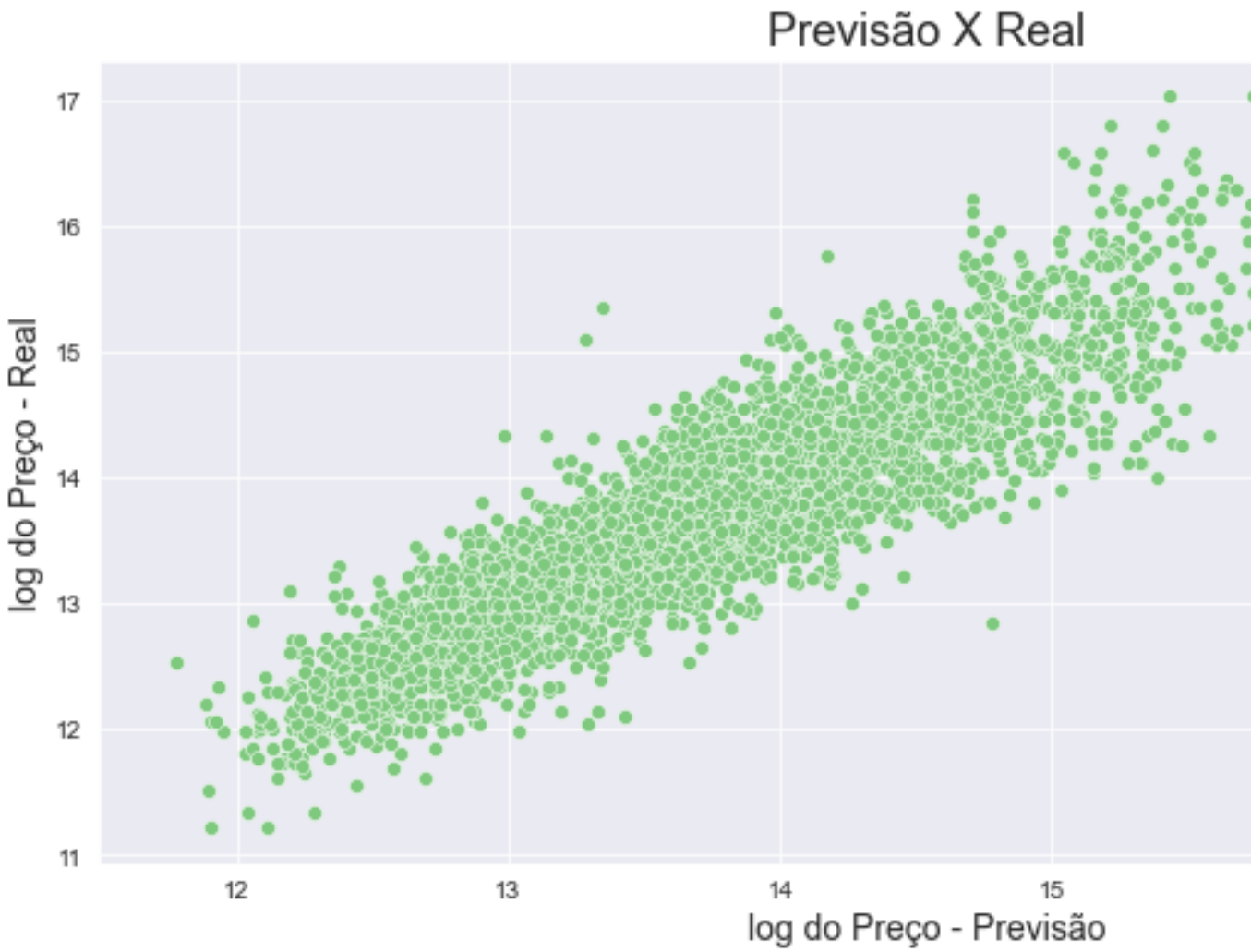
In [69]:
```
y_previsto_train = modelo.predict(X_train)
```

Gráfico de dispersão entre valor estimado e valor real

https://seaborn.pydata.org/generated/seaborn.scatterplot.html

In [70]:
```
ax = sns.scatterplot(x=y_previsto_train, y=y_train)
ax.figure.set_size_inches(12, 6)
ax.set_title('Previsão X Real', fontsize=18)
ax.set_xlabel('log do Preço - Previsão', fontsize=14)
ax.set_ylabel('log do Preço - Real', fontsize=14)
ax
```

Out[70]:
```
<AxesSubplot:title={'center':'Previsão X Real'}, xlabel='log do Preço
- Previsão', ylabel='log do Preço - Real'>
```



Obtendo os resíduos

In [71]:

```
residuo = y_train - y_previsto_train
```

Plotando a distribuição de frequências dos resíduos

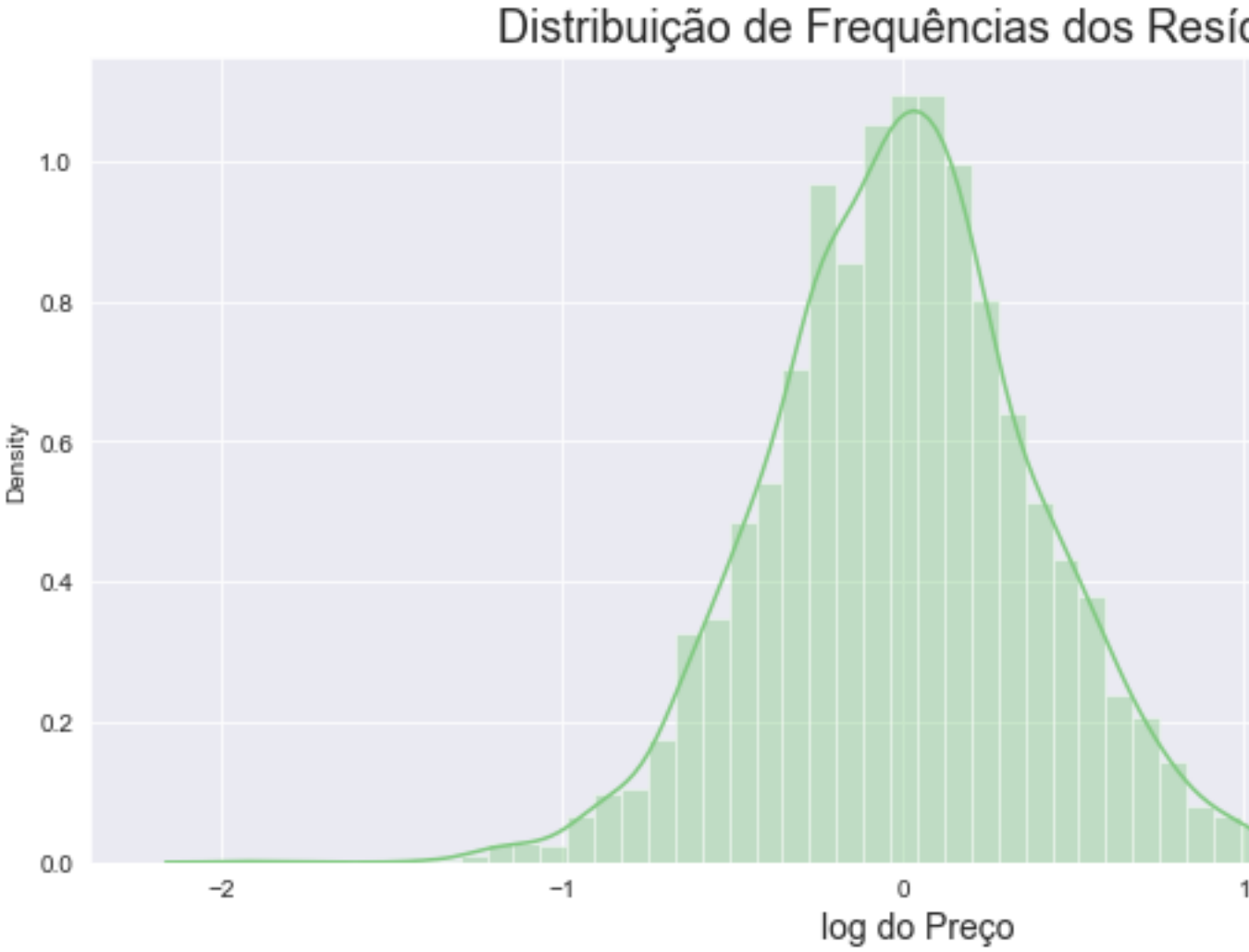
In [72]:

```
ax = sns.distplot(residuo)
ax.figure.set_size_inches(12, 6)
ax.set_title('Distribuição de Frequências dos Resíduos', fontsize=18)
ax.set_xlabel('log do Preço', fontsize=14)
ax
C:\Users\EDWARD SOUSA\anaconda3\lib\site-
packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with
```

```
similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

Out[72]:

```
<AxesSubplot:title={'center':'Distribuição de Frequências dos
Resíduos'}, xlabel='log do Preço', ylabel='Density'>
```



In [ ]: