In [1]:

```
import pandas as pd
import numpy as np

from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import VarianceThreshold
from sklearn.decomposition import PCA
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
%matplotlib notebook
```

In [2]:

```
data = pd.read_csv('./googleplaystore.csv')
data = data.assign(Installs_numeric=0)
data.head(3)
```

Out[2]:

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	De
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	

In [3]:

```
print('%d Total amostras'%len(data))
data = data.dropna()
print('%d Total amostras (s/ NaN)'%len(data))
```

```
10841 Total amostras
9360 Total amostras (s/ NaN)
```

In [4]:

```
data.groupby(['Category']).agg('count')
category = {'1.9': 0, 'ART_AND_DESIGN': 1, 'AUTO_AND_VEHICLES': 2, 'BEAUTY': 3, 'BOOKS_AND_
data['categoryID'] = data['Category'].map(lambda i: category[i])
```

In [5]:

```
data.groupby(['Type']).agg('count')
type = {'0': 0, 'Free': 1, 'Paid': 2 }
data['typeID'] = data['Type'].map(lambda i: type[i])
```

In [6]:

```
data.groupby(['Content Rating']).agg('count')
type = {'Adults only 18+': 0, 'Everyone': 1, 'Everyone 10+': 2, 'Mature 17+': 3, 'Teen': 4,
data['contentRatingID'] = data['Content Rating'].map(lambda i: type[i])
```

In [7]:

```
data['installsNumeric'] = data['Installs'].map(lambda i: int(i.replace('+','').replace(',',
```

In [8]:

```
data['priceNumeric'] = data['Price'].map(lambda i: float(i.replace('$','').replace('.','))
```

In [9]:

```
data['reviewsNumeric'] = data['Reviews'].map(lambda i: int(i))
```

In [10]:

```
coluns_x = ['categoryID', 'Rating', 'reviewsNumeric', 'typeID', 'priceNumeric', 'contentRat
coluns_y = ['installsNumeric']

data_x = data[coluns_x]
data_y = data[coluns_y]

train_x, test_x, train_y, test_y = train_test_split(data_x.values,
                                                    data_y.values,
                                                    test_size=.3)
```

In [11]:

```
model = Pipeline([('norm', MinMaxScaler()),
                  ('var_thr', VarianceThreshold(threshold=.02)),
                  ('pca', PCA(n_components=3)),
                  ('regressor', LinearRegression())
                 ])
model.fit(train_x, train_y)
```

Out[11]:

```
Pipeline(memory=None,
     steps=[('norm', MinMaxScaler(copy=True, feature_range=(0, 1))), ('var_t
hr', VarianceThreshold(threshold=0.02)), ('pca', PCA(copy=True, iterated_pow
er='auto', n_components=3, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)), ('regressor', LinearRegression
(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False))])
```

In [12]:

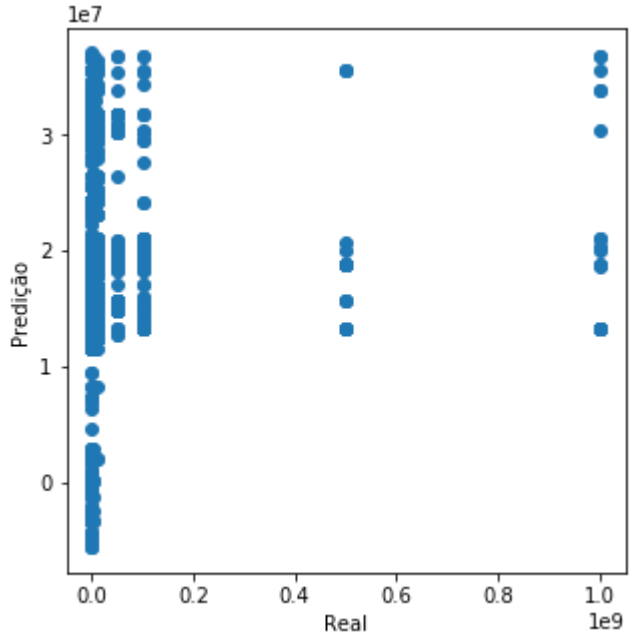
```
predictions = model.predict(test_x)
predictions
```

Out[12]:

```
array([[14508843.99645712],
       [17729400.11596472],
       [12752177.02218025],
       ...,
       [ -693379.90039856],
       [30970952.43433093],
       [12459399.19313411]])
```

In [13]:

```
plt.figure(figsize=(5,5))
plt.xlabel("Real")
plt.ylabel("Predição")
plt.plot([0,1],[0,1],c='k')
plt.scatter(test_y, predictions);
```



In [14]:

```
print("MAE: %.3f"%(mean_absolute_error(test_y, predictions)))
```

```
MAE: 29070431.979
```

In [ ]:

```
# Conclusões
# Foi escolhido um Dataset de downloads de Apps na Play Store
# Usando CRISP-DM bscamos resolver um problema conhecido, criando um modelo que
# confrontasse os dados com o Número de downloads, utilizando uma técnica de regressão para
# Foi necessário normalizar quase todos os campos, pois quase todos eram Strings com símbol
# O resultado foi um gráfico vertifical, com quase todos os valores variando entre 0 e 0.1
# Ao chegar nesse resultado, realizamos uma nova iteração e tentamos mudar nossa analise de
# no intuito de analisar esses dados em forma de clusters, porém tivemos problemas na exibi
# voltar para o resultado inicial.
```