Previsão de Votação de Deputados

Aluno: Gustavo Silva Medeiros

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib

import matplotlib.pyplot as plt
from scipy.stats import skew
from scipy.stats.stats import pearsonr
from sklearn import preprocessing

%config InlineBackend.figure_format = 'png'
%matplotlib inline
```

2) Considere o pipeline mostrado nesse link para construir seus modelos de regressão. Isso implica, dentre outras coisas:

2.1) Analisar as distribuições das variáveis para ver se estão enviesadas e precisam de correção; tratamento de valores ausentes, variáveis categóricas e normalização, quando for o caso.

In [2]:
```
# Colunas carregadas do CSV
cols = [
    "ano",
    "nome",
    "sexo",
    "estado_civil",
    "grau",
    "ocupacao",
    "uf",
    "partido",
    "quantidade_doacoes",
    "quantidade_doadores",
    "total_receita",
    "media_receita",
    "recursos_de_outros_candidatos/comites",
    "recursos_de_pessoas_fisicas",
    "recursos_de_pessoas_juridicas",
    "recursos_proprios",
    "quantidade_despesas",
    "quantidade_fornecedores",
    "total_despesa",
    "media_despesa",
    "votos",
]
# OBS 1: As colunas "sequencial_candidato" e "cargo" não foram consideradas atribu

df = pd.read_csv("eleicoes_2006_a_2010.csv", usecols=cols, index_col="nome")

# separa os dados de 2006 como treino e os dados de 2010 como teste e remove a colu
train = df[(df['ano'] == 2006)].loc[:,'uf':]
test = df[(df['ano'] == 2010)].loc[:,'uf':]
```

```
# junta os dados de 2006 e 2010 sem as colunas "ano" e "votos"
all_data = pd.concat((
    train.loc[:,:'ocupacao'],
    test.loc[:,:'ocupacao']
))
```

In [3]:
```
all_data.head()
```

Out[3]:

nome	uf	partido	quantidade_doacoes	quantidade_doadores	total_receita	media_receita
JOSÉ LUIZ NOGUEIRA DE SOUSA	AP	PT	6	6	16600.00	2766.666667
LOIVA DE OLIVEIRA	RO	PT	13	13	22826.00	1755.846154
MARIA DALVA DE SOUZA FIGUEIREDO	AP	PT	17	16	148120.80	9257.550000
ROMALDO MILANI	MS	PRONA	6	6	3001.12	500.186667
ANSELMO DE JESUS ABREU	RO	PT	48	48	NaN	NaN

In [4]:
```
train.head()
```

Out[4]:

nome	uf	partido	quantidade_doacoes	quantidade_doadores	total_receita	media_receita
JOSÉ LUIZ NOGUEIRA DE SOUSA	AP	PT	6	6	16600.00	2766.666667
LOIVA DE OLIVEIRA	RO	PT	13	13	22826.00	1755.846154
MARIA DALVA DE SOUZA FIGUEIREDO	AP	PT	17	16	148120.80	9257.550000
ROMALDO MILANI	MS	PRONA	6	6	3001.12	500.186667
ANSELMO DE JESUS ABREU	RO	PT	48	48	NaN	NaN

In [5]:
```
test.head()
```

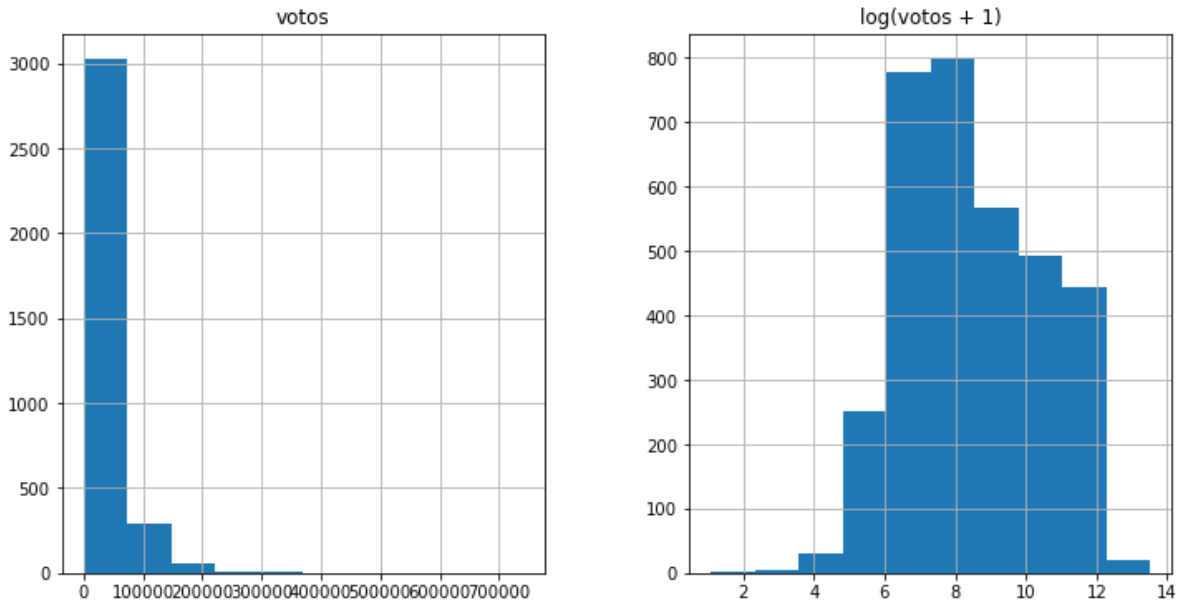
Out[5]:

nome	uf	partido	quantidade_doacoes	quantidade_doadores	total_receita	media_receita
ANTONIA LUCILEIA CRUZ RAMOS CAMARA	AC	PSC	36	35	406891.92	11625.483429
DEODATO NUNES DE FRANÇA	AC	PMDB	3	3	6990.00	2330.000000
EDSON FIRMINO DE PAULA	AC	PSDB	3	3	1840.00	613.333333
ELISABETH APARECIDA GARCIA RODRIGUES	AC	PSDB	1	1	440.00	440.000000
FLAVIANO FLAVIO BAPTISTA DE MELO	AC	PMDB	15	13	241500.00	18576.923077

Data preprocessing

In [6]:
```
matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)
prices = pd.DataFrame({"votos":train["votos"], "log(votos + 1)":np.log1p(train["vo
prices.hist()
```

Out[6]:
```
array([[<AxesSubplot:title={'center':'votos'}>,
        <AxesSubplot:title={'center':'log(votos + 1)'}>]], dtype=object)
```



In [7]:
```
# log transform the target:
train["votos"] = np.log1p(train["votos"])

# log transform skewed numeric features:
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index
```

```
skewed_feats = train[numeric_feats].apply(lambda x: skew(x.dropna())) #compute skew
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index

all_data[skewed_feats] = np.log1p(all_data[skewed_feats])
```

In [8]:
```
all_data = pd.get_dummies(all_data)
all_data.head()
```

Out[8]:

nome	quantidade_doacoes	quantidade_doadores	total_receita	media_receita	recursos_de_
JOSÉ LUIZ NOGUEIRA DE SOUSA	1.945910	1.945910	9.717218	7.925760	
LOIVA DE OLIVEIRA	2.639057	2.639057	10.035699	7.471276	
MARIA DALVA DE SOUZA FIGUEIREDO	2.890372	2.833213	11.905790	9.133303	
ROMALDO MILANI	1.945910	1.945910	8.007074	6.216979	
ANSELMO DE JESUS ABREU	3.891820	3.891820	NaN	NaN	

5 rows × 259 columns

In [9]:
```
# filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())
all_data.head()
```

Out[9]:

nome	quantidade_doacoes	quantidade_doadores	total_receita	media_receita	recursos_de_
JOSÉ LUIZ NOGUEIRA DE SOUSA	1.945910	1.945910	9.717218	7.925760	
LOIVA DE OLIVEIRA	2.639057	2.639057	10.035699	7.471276	
MARIA DALVA DE SOUZA FIGUEIREDO	2.890372	2.833213	11.905790	9.133303	
ROMALDO MILANI	1.945910	1.945910	8.007074	6.216979	
ANSELMO DE JESUS ABREU	3.891820	3.891820	9.634619	7.466020	

5 rows × 259 columns

2.2) Construir modelos de regressão com (ridge e lasso) e sem regularização.

2.3) Considerar outros modelos ainda não vistos em sala de sua escolha (e.g. SVR, Regression Trees, KNN e Random Florests).

2.4) Tunar os hiperâmetros para cada caso e retornar os rmses de validação cruzada para todos os modelos avaliados.

2.5) Plotar os resíduos versus predições e analisar se esses plots representam bons indícios da adequabilidade dos modelos a esse problema.

In [10]:
```
# creating matrices for sklearn:
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.votos
```

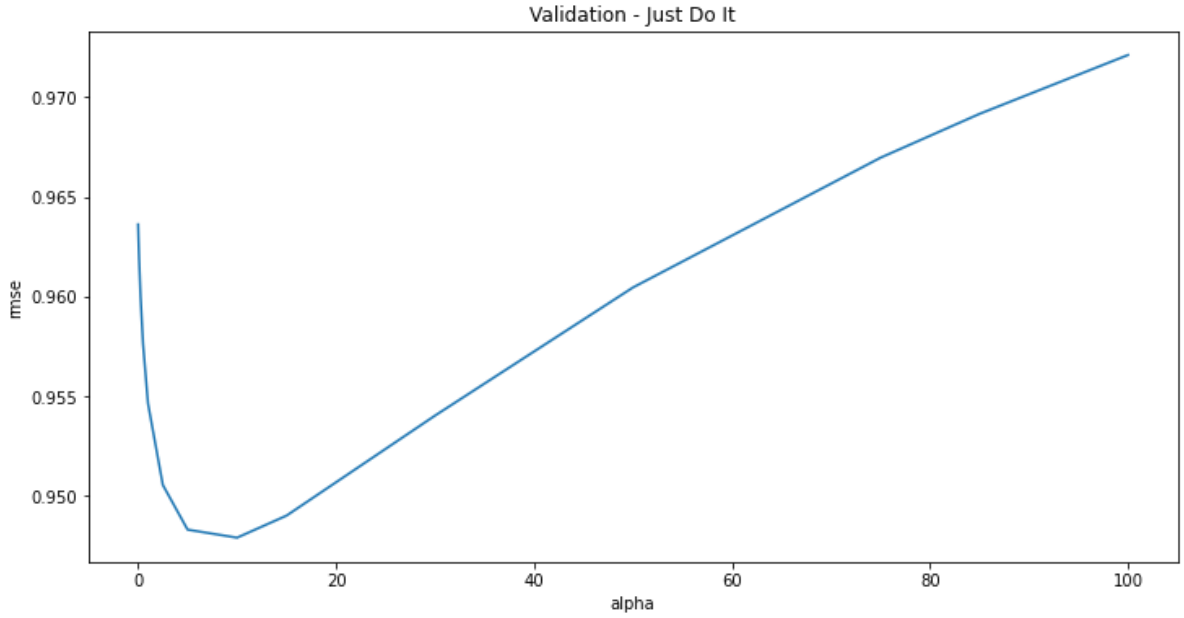
In [11]:
```
from sklearn.linear_model import Ridge, RidgeCV, ElasticNet, LassoCV, LassoLarsCV
from sklearn.model_selection import cross_val_score

def rmse_cv(model, X_train, y):
    rmse= np.sqrt(-cross_val_score(model, X_train, y, scoring="neg_mean_squared_er
    return(rmse)
```

Ridge

In [12]:
```
model_ridge = None

# tunando hiperâmetros
alphas = [0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 1

cv_ridge = []
min_ridge_rmse = float("inf")
for alpha in alphas:
    model_ridge_tmp = Ridge(alpha = alpha)
    ridge_rmse = rmse_cv(model_ridge_tmp, X_train, y).mean()
    cv_ridge.append(ridge_rmse)
    if ridge_rmse < min_ridge_rmse:
        min_ridge_rmse = ridge_rmse
        model_ridge = model_ridge_tmp
```

In [13]:
```
cv_ridge = pd.Series(cv_ridge, index = alphas)
cv_ridge.plot(title = "Validation - Just Do It")
plt.xlabel("alpha")
plt.ylabel("rmse")
```

Out[13]:
```
Text(0, 0.5, 'rmse')
```



Ridge: RMSE

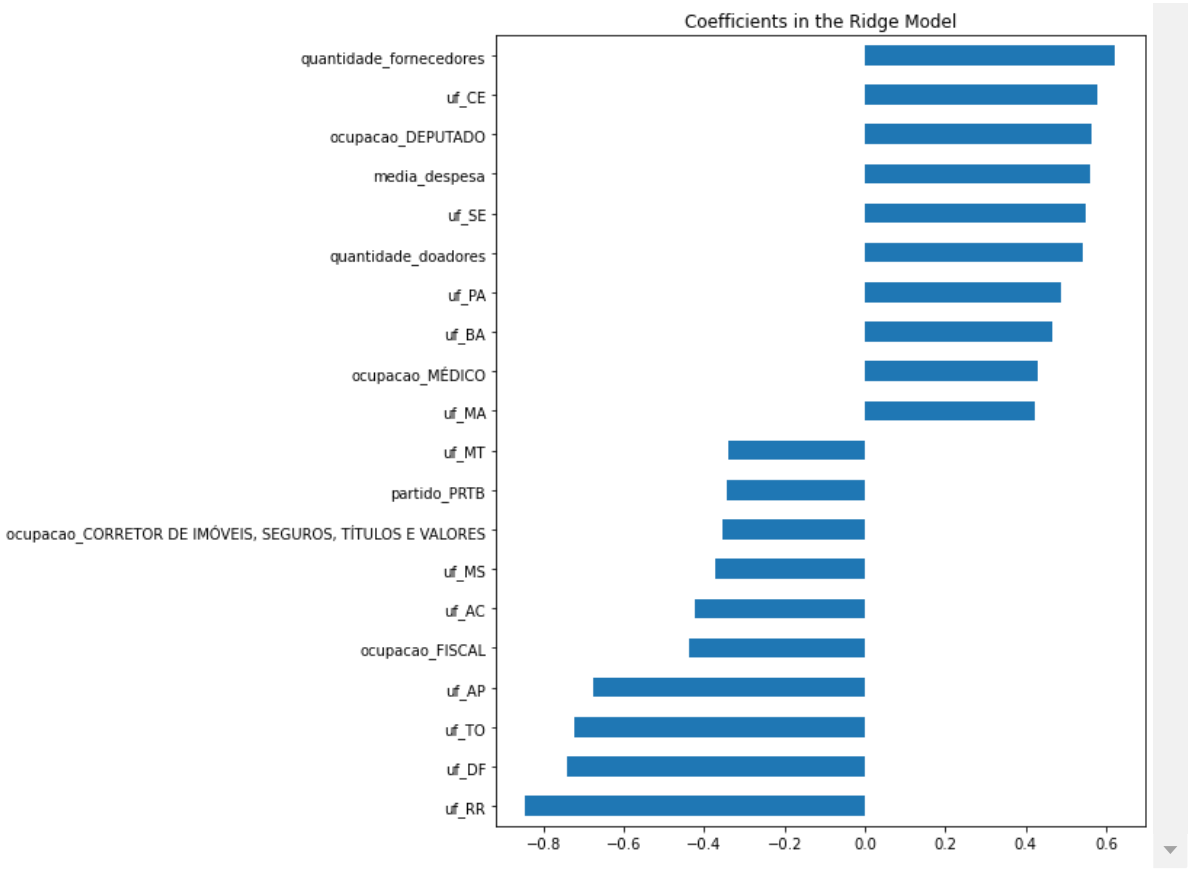
In [14]:
```
min_ridge_rmse
```

Out[14]:
```
0.9479115249847091
```

Ridge: Coefficients

In [15]:
```
model_ridge.fit(X_train, y)
coef = pd.Series(model_ridge.coef_, index = X_train.columns)
imp_coef = pd.concat([coef.sort_values().head(10),
                      coef.sort_values().tail(10)])

matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Ridge Model")
```

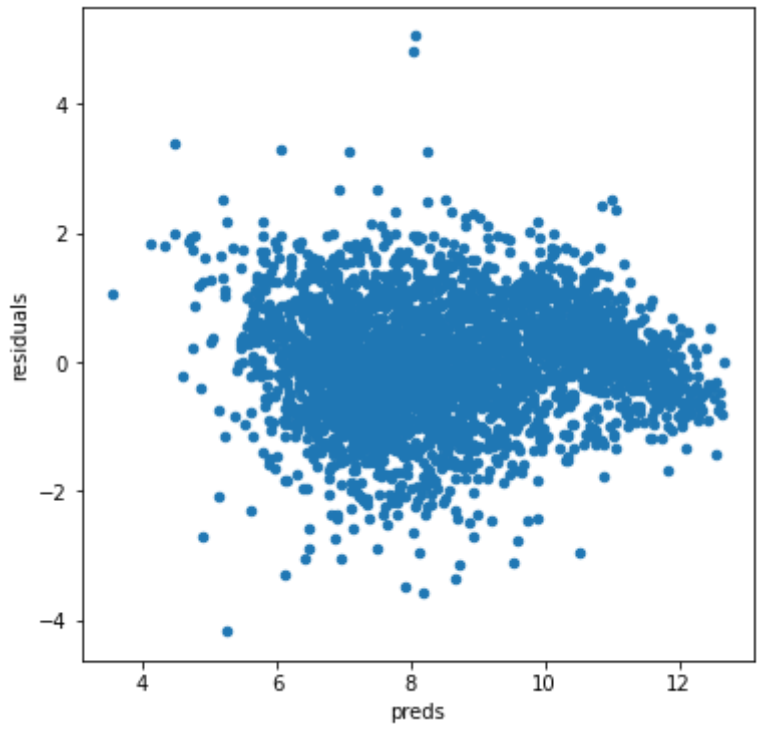
Out[15]:
```
Text(0.5, 1.0, 'Coefficients in the Ridge Model')
```



Ridge: Resíduos vs Predições

In [16]:
```
# let's look at the residuals as well:
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)

preds = pd.DataFrame({"preds":model_ridge.predict(X_train), "true":y})
preds["residuals"] = preds["true"] - preds["preds"]
preds.plot(x = "preds", y = "residuals",kind = "scatter")
```

Out[16]:
```
<AxesSubplot:xlabel='preds', ylabel='residuals'>
```



O gráfico de resíduos versus predições do modelo Ridge parece bom. Os dados não estão

muito espalhados.

Lasso

In [17]:
```
# tunando hiperâmetros
model_lasso = LassoCV(alphas = [1, 0.75, 0.5, 0.25, 0.1, 0.075, 0.05, 0.025, 0.01,
```

Lasso: RMSE

In [18]:
```
rmse_cv(model_lasso, X_train, y).mean()
```

Out[18]:
```
0.9585032441908858
```

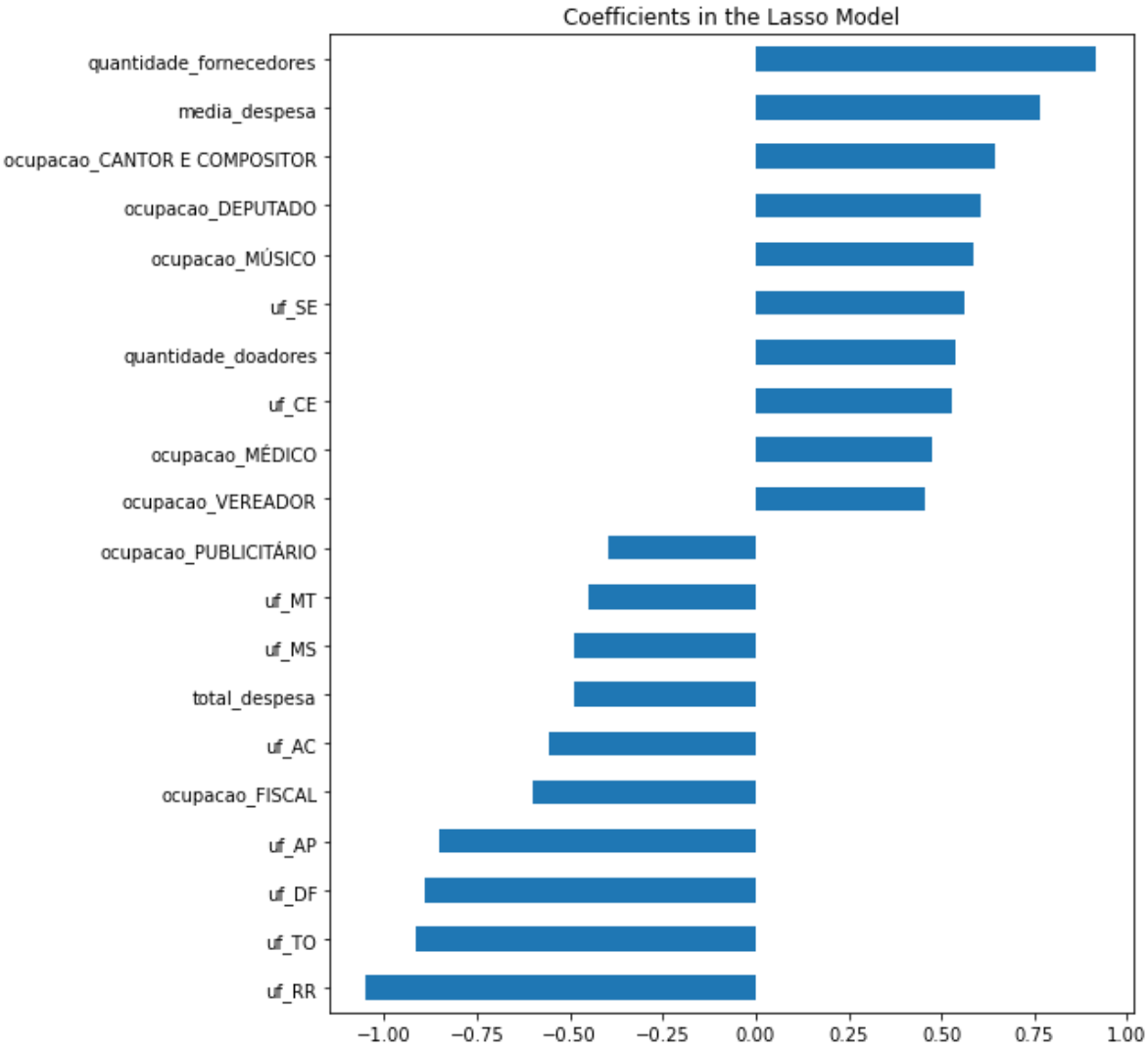
Lasso: Coefficients

In [19]:
```
coef = pd.Series(model_lasso.coef_, index = X_train.columns)
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other
```

```
Lasso picked 111 variables and eliminated the other 148 variables
```

In [20]:
```
imp_coef = pd.concat([coef.sort_values().head(10),
                      coef.sort_values().tail(10)])

matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("Coefficients in the Lasso Model")
```

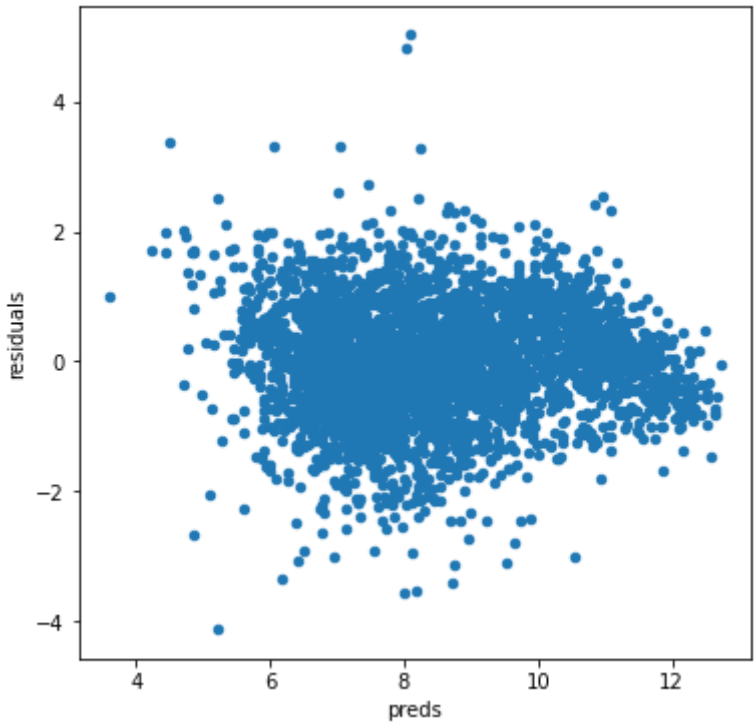
Out[20]:
```
Text(0.5, 1.0, 'Coefficients in the Lasso Model')
```



Lasso: Resíduos vs Predições

In [21]:
```
# let's look at the residuals as well:
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)

preds = pd.DataFrame({"preds":model_lasso.predict(X_train), "true":y})
preds["residuals"] = preds["true"] - preds["preds"]
preds.plot(x = "preds", y = "residuals",kind = "scatter")
```

Out[21]:
```
<AxesSubplot:xlabel='preds', ylabel='residuals'>
```



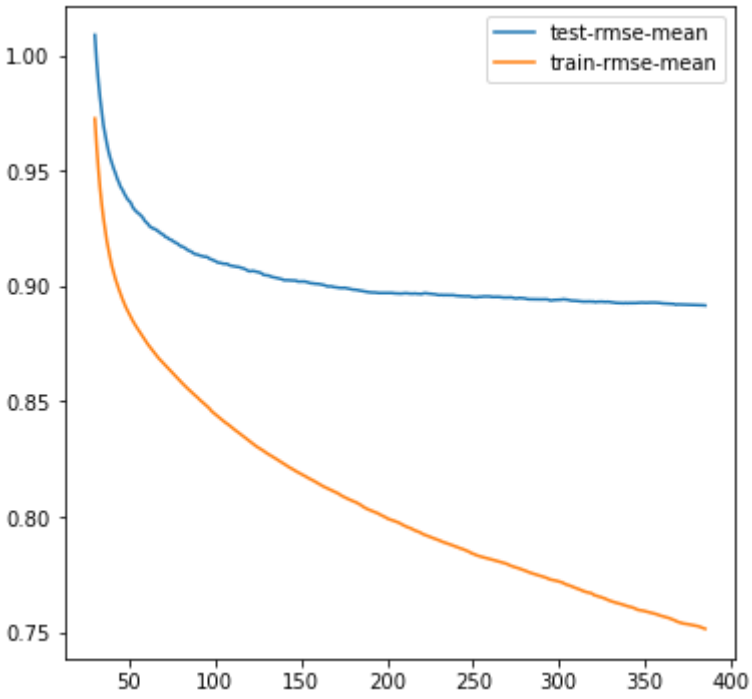
O gráfico de resíduos versus predições do modelo Lasso também parece bom. A maioria dos dados não estão muito espalhados.

In [22]:
```
import xgboost as xgb

dtrain = xgb.DMatrix(X_train, label = y)
dtest = xgb.DMatrix(X_test)

params = {"max_depth":2, "eta":0.1}
model = xgb.cv(params, dtrain,  num_boost_round=500, early_stopping_rounds=100)
```

```
C:\Users\ghust\anaconda3\lib\site-packages\xgboost\compat.py:36: FutureWarning: pa
ndas.Int64Index is deprecated and will be removed from pandas in a future version.
Use pandas.Index with the appropriate dtype instead.
  from pandas import MultiIndex, Int64Index
C:\Users\ghust\anaconda3\lib\site-packages\xgboost\data.py:250: FutureWarning: pan
das.Int64Index is deprecated and will be removed from pandas in a future version.
Use pandas.Index with the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):
```

In [23]:
```
model.loc[30:,["test-rmse-mean", "train-rmse-mean"]].plot()
```

Out[23]:
```
<AxesSubplot:>
```



In [24]:
```
model_xgb = xgb.XGBRegressor(n_estimators=360, max_depth=2, learning_rate=0.1) #th
model_xgb.fit(X_train, y)
```

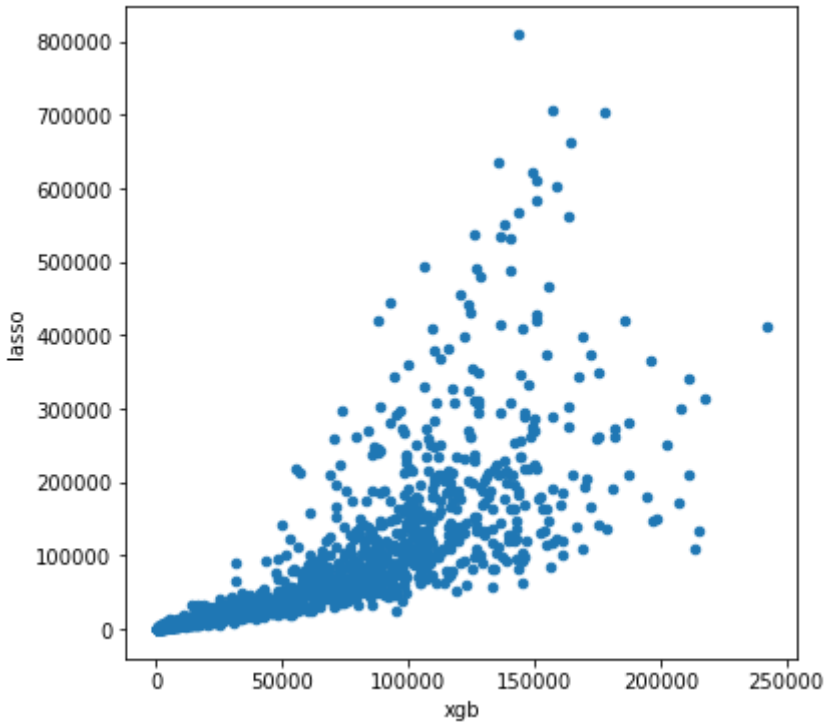
```
C:\Users\ghust\anaconda3\lib\site-packages\xgboost\data.py:250: FutureWarning: pan
das.Int64Index is deprecated and will be removed from pandas in a future version.
Use pandas.Index with the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):
```

Out[24]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.1, max_delta_step=0,
             max_depth=2, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=360, n_jobs=16,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [25]:
```
xgb_preds = np.expm1(model_xgb.predict(X_test))
lasso_preds = np.expm1(model_lasso.predict(X_test))
```

In [26]:
```
from scipy import stats

predictions = pd.DataFrame({"xgb":xgb_preds, "lasso":lasso_preds})

# remove outliers
predictions = predictions[(np.abs(stats.zscore(predictions)) < 10).all(axis=1)]

predictions.plot(x = "xgb", y = "lasso", kind = "scatter")
```

Out[26]:
```
<AxesSubplot:xlabel='xgb', ylabel='lasso'>
```



In [27]:
```
preds = 0.7 * lasso_preds + 0.3 * xgb_preds

# converte os votos para int
preds = preds.astype('int')
```

In [28]:
```
solution = pd.DataFrame({"nome":test.index, "votos":preds})
solution.to_csv("ridge_sol.csv", index = False)
```

In [29]:
```
solution.head(20)
```

Out[29]:

	nome	votos
0	ANTONIA LUCILEIA CRUZ RAMOS CAMARA	26239
1	DEODATO NUNES DE FRANÇA	1634
2	EDSON FIRMINO DE PAULA	734
3	ELISABETH APARECIDA GARCIA RODRIGUES	360
4	FLAVIANO FLAVIO BAPTISTA DE MELO	26706
5	FRANCISCO ALVES VIEIRA	1054
6	FRANCISCO CARLOS DE OLIVEIRA DE LIMA	783
7	FRANCISCO JOSE BENICIO DIAS	897
8	JOSE ALVES DE MORAES	519
9	MARCIO MIGUEL BITTAR	22351
10	MARIA DA ROCHA SEVERO	300
11	MAURICIO DINIZ LIMA	509
12	MIGUEL ARCANJO DE SOUZA CARNEIRO	876
13	RAIMUNDO NONATO DE CASTRO	2168
14	SOLANGE MARIA PINHO PASCOAL	5279
15	UAGLA BELMONT ALVES	1565
16	EDILBERTO AFONSO DE MORAES JUNIOR	18604
17	FERNANDO MELO DA COSTA	27458
18	GEHLEN DINIZ ANDRADE	3809
19	GILSON GOMES DE OLIVEIRA	4451

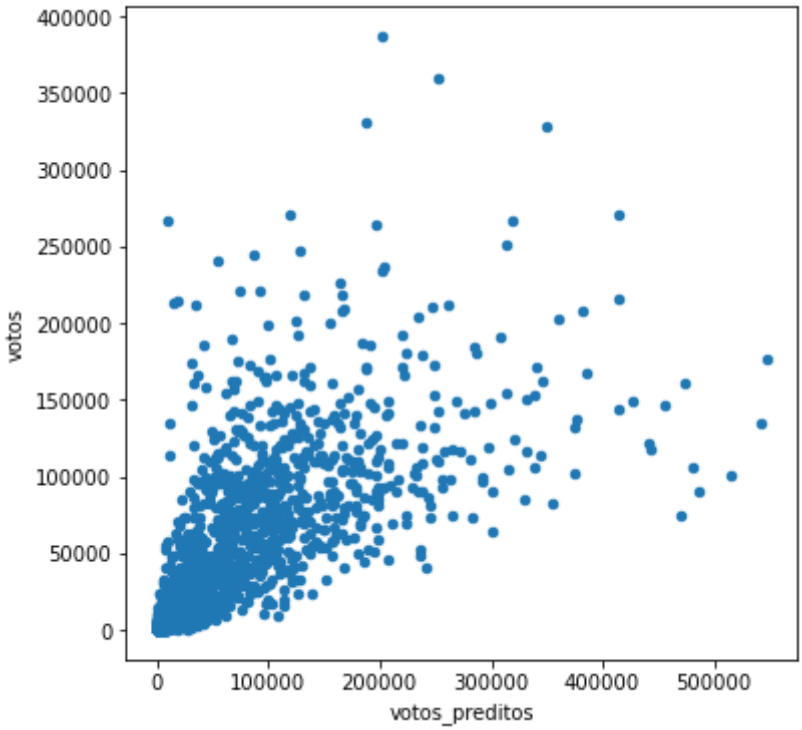
In [30]:
```
test.loc[:, "votos":"votos"].head(20)
```

Out[30]:

nome	votos
ANTONIA LUCILEIA CRUZ RAMOS CAMARA	15849
DEODATO NUNES DE FRANÇA	4620
EDSON FIRMINO DE PAULA	312
ELISABETH APARECIDA GARCIA RODRIGUES	1357
FLAVIANO FLAVIO BAPTISTA DE MELO	36301
FRANCISCO ALVES VIEIRA	3378
FRANCISCO CARLOS DE OLIVEIRA DE LIMA	3377
FRANCISCO JOSE BENICIO DIAS	1082
JOSE ALVES DE MORAES	943
MARCIO MIGUEL BITTAR	52183
MARIA DA ROCHA SEVERO	113
MAURICIO DINIZ LIMA	744
MIGUEL ARCANJO DE SOUZA CARNEIRO	567
RAIMUNDO NONATO DE CASTRO	2492
SOLANGE MARIA PINHO PASCOAL	10513
UAGLA BELMONT ALVES	452
EDILBERTO AFONSO DE MORAES JUNIOR	149
FERNANDO MELO DA COSTA	11018
GEHLEN DINIZ ANDRADE	2944
GILSON GOMES DE OLIVEIRA	707

In [31]:
```
votos_vs_votos_preditos = pd.DataFrame({"votos": test["votos"].values, "votos_pred

# remove outliers
votos_vs_votos_preditos = votos_vs_votos_preditos[(np.abs(stats.zscore(votos_vs_vo

votos_vs_votos_preditos.plot(x = "votos_preditos", y = "votos", kind = "scatter")
```

Out[31]:
```
<AxesSubplot:xlabel='votos_preditos', ylabel='votos'>
```



RandomForestRegressor

In [32]:
```
y_test = test.votos

from sklearn.ensemble import RandomForestRegressor

model_rf = RandomForestRegressor(n_estimators = 50, random_state = 42, max_depth=8

# Train the model on training data
model_rf.fit(X_train, y);
```

In [33]:
```
# Use the forest's predict method on the test data
predictions = model_rf.predict(X_test)

# Calculate the absolute errors
errors = abs(predictions - y_test)

# Print out the mean absolute error (mae)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
```

```
Mean Absolute Error: 22454.66 degrees.
```

In [34]:
```
# Calculate mean absolute percentage error (MAPE)
mape = 100 * (errors / y_test)

# Calculate and display accuracy
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

```
Accuracy: 0.94 %.
```

RandomForestRegressor: RMSE

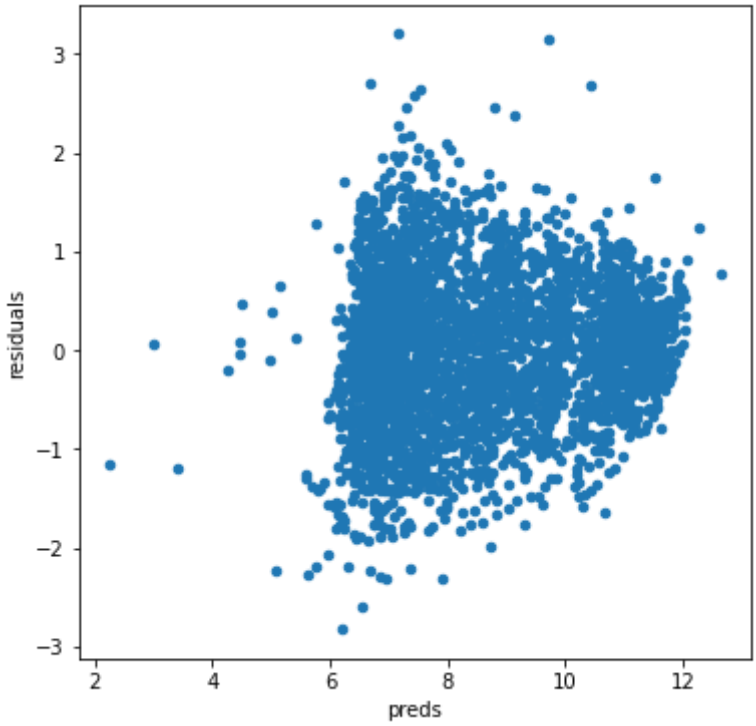
In [35]:
```
rmse_cv(model_rf, X_train, y).mean()
```

Out[35]:
```
0.9399565049950297
```

RandomForestRegressor: Resíduos vs Predições

In [36]:
```
# let's look at the residuals as well:
matplotlib.rcParams['figure.figsize'] = (6.0, 6.0)

preds = pd.DataFrame({"preds":model_rf.predict(X_train), "true":y})
preds["residuals"] = preds["true"] - preds["preds"]
preds.plot(x = "preds", y = "residuals",kind = "scatter")
```

Out[36]:
```
<AxesSubplot:xlabel='preds', ylabel='residuals'>
```



O gráfico de resíduos versus predições do modelo Ramdom Forest aparenta que os dados não estão muito espalhados.

3) Alguns dias antes da entrega final serão liberados os dados de teste referentes à 2014 para validação final dos seus melhores modelos.

3.1) Dica: Uma coisa que você pode fazer é usar os dados de 2006 como treino e os de 2010 como validação. Uma vez encontrados os melhores modelos para 2010 junte 2006+2010, retreine, e aplique o modelo aos dados de 2014 que serão liberados.

In [37]:
```
# Colunas carregadas do CSV
cols = [
    "ano",
    "nome",
    "sexo",
    "estado_civil",
    "grau",
    "ocupacao",
    "uf",
    "partido",
    "quantidade_doacoes",
    "quantidade_doadores",
```

```
    "total_receita",
    "media_receita",
    "recursos_de_outros_candidatos/comites",
    "recursos_de_pessoas_fisicas",
    "recursos_de_pessoas_juridicas",
    "recursos_proprios",
    "quantidade_despesas",
    "quantidade_fornecedores",
    "total_despesa",
    "media_despesa",
    "votos",
]

df_2006_2010 = pd.read_csv("eleicoes_2006_a_2010.csv", usecols=cols, index_col="nor

# junta os dados de 2006 e 2010 para treino
train = df_2006_2010.loc[:,'uf':]

# dados de 2014 como teste
df_2014 = pd.read_csv("eleicoes_2014.csv", usecols=cols, index_col="nome")
test = df_2014.loc[:,'uf':]

# junta os dados de 2006, 2010 e 2014 sem as colunas "ano" e "votos"
all_data = pd.concat((
    train.loc[:,:'ocupacao'],
    test.loc[:,:'ocupacao']
))
```

In [38]:
```
train.head()
```

Out[38]:

nome	uf	partido	quantidade_doacoes	quantidade_doadores	total_receita	media_receita
JOSÉ LUIZ NOGUEIRA DE SOUSA	AP	PT	6	6	16600.00	2766.666667
LOIVA DE OLIVEIRA	RO	PT	13	13	22826.00	1755.846154
MARIA DALVA DE SOUZA FIGUEIREDO	AP	PT	17	16	148120.80	9257.550000
ROMALDO MILANI	MS	PRONA	6	6	3001.12	500.186667
ANSELMO DE JESUS ABREU	RO	PT	48	48	NaN	NaN

In [39]:
```
test.head()
```

Out[39]:

nome	uf	partido	quantidade_doacoes	quantidade_doadores	total_receita	media_receita
EMERSON DA SILVA SANTOS	AC	PSOL	3	3	1580.00	526.666667
GERALDO SILVA DOS SANTOS	AC	PSOL	5	5	3180.00	636.000000
CARLOS CESAR CORREIA DE MESSIAS	AC	PSB	40	38	333293.13	8770.871842
IDESIO LUIS FRANKE	AC	PT	29	29	156719.32	5404.114483
LEONARDO CUNHA DE BRITO	AC	PT	160	146	711083.00	4870.431507

In [40]:
```
all_data
```

Out[40]:

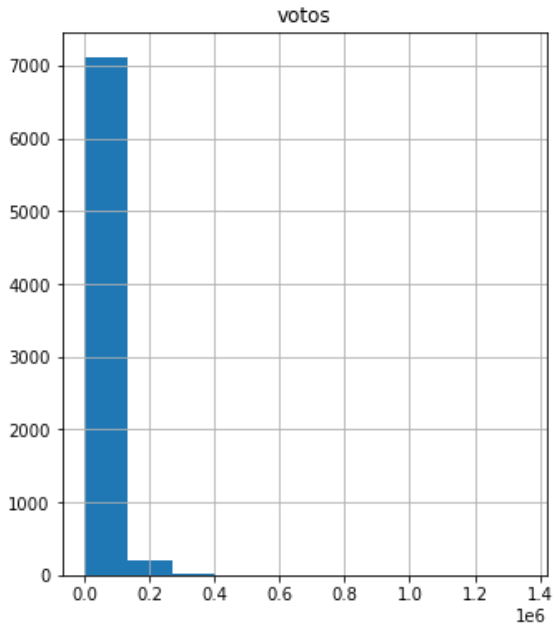
nome	uf	partido	quantidade_doacoes	quantidade_doadores	total_receita	media_receita
JOSÉ LUIZ NOGUEIRA DE SOUSA	AP	PT	6	6	16600.00	2766.666667
LOIVA DE OLIVEIRA	RO	PT	13	13	22826.00	1755.846154
MARIA DALVA DE SOUZA FIGUEIREDO	AP	PT	17	16	148120.80	9257.550000
ROMALDO MILANI	MS	PRONA	6	6	3001.12	500.186667
ANSELMO DE JESUS ABREU	RO	PT	48	48	NaN	NaN
...	...	...	...	...	...	...
JOENICE PEREIRA RIBEIRO	TO	PR	7	6	6334.29	1055.715000
TIAGO DE PAULA ANDRINO	TO	PP	42	42	1738508.82	41393.067143
ETEVALDO DA PAZ NONATO	TO	PSOL	3	3	1230.00	410.000000
IVARDO SANTANA	TO	PSOL	2	2	900.00	450.000000
MARIA LUCIA SOARES VIANA	TO	PSOL	10	9	9095.00	1010.555556

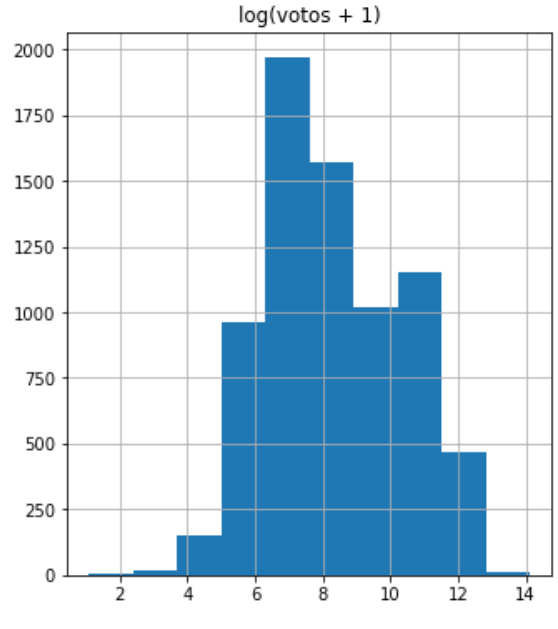
12266 rows × 18 columns

Data preprocessing

In [41]:
```
matplotlib.rcParams['figure.figsize'] = (12.0, 6.0)
prices = pd.DataFrame({"votos":train["votos"], "log(votos + 1)":np.log1p(train["vo
prices.hist()
```

Out[41]:
```
array([[<AxesSubplot:title={'center':'votos'}>,
        <AxesSubplot:title={'center':'log(votos + 1)'}>]], dtype=object)
```





In [42]:
```
# log transform the target:
train["votos"] = np.log1p(train["votos"])

# log transform skewed numeric features:
numeric_feats = all_data.dtypes[all_data.dtypes != "object"].index

skewed_feats = train[numeric_feats].apply(lambda x: skew(x.dropna())) #compute skew
skewed_feats = skewed_feats[skewed_feats > 0.75]
skewed_feats = skewed_feats.index

all_data[skewed_feats] = np.log1p(all_data[skewed_feats])
```

In [43]:
```
all_data = pd.get_dummies(all_data)
all_data.head()
```

Out[43]:

nome	quantidade_doacoes	quantidade_doadores	total_receita	media_receita	recursos_de_(
JOSÉ LUIZ NOGUEIRA DE SOUSA	1.945910	1.945910	9.717218	7.925760	
LOIVA DE OLIVEIRA	2.639057	2.639057	10.035699	7.471276	
MARIA DALVA DE SOUZA FIGUEIREDO	2.890372	2.833213	11.905790	9.133303	
ROMALDO MILANI	1.945910	1.945910	8.007074	6.216979	
ANSELMO DE JESUS ABREU	3.891820	3.891820	NaN	NaN	

5 rows × 281 columns

In [44]:
```
# filling NA's with the mean of the column:
all_data = all_data.fillna(all_data.mean())
all_data.head()
```

Out[44]:

nome	quantidade_doacoes	quantidade_doadores	total_receita	media_receita	recursos_de_
JOSÉ LUIZ NOGUEIRA DE SOUSA	1.945910	1.945910	9.717218	7.925760	
LOIVA DE OLIVEIRA	2.639057	2.639057	10.035699	7.471276	
MARIA DALVA DE SOUZA FIGUEIREDO	2.890372	2.833213	11.905790	9.133303	
ROMALDO MILANI	1.945910	1.945910	8.007074	6.216979	
ANSELMO DE JESUS ABREU	3.891820	3.891820	9.670482	7.468538	

5 rows × 281 columns

In [45]:
```
# creating matrices for sklearn:
X_train = all_data[:train.shape[0]]
X_test = all_data[train.shape[0]:]
y = train.votos
```

Ridge (retreino)

In [46]:
```
model_ridge = None

alphas = [0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 1

cv_ridge = []
min_ridge_rmse = float("inf")
for alpha in alphas:
    model_ridge_tmp = Ridge(alpha = alpha)
    ridge_rmse = rmse_cv(model_ridge_tmp, X_train, y).mean()
    cv_ridge.append(ridge_rmse)
    if ridge_rmse < min_ridge_rmse:
        min_ridge_rmse = ridge_rmse
        model_ridge = model_ridge_tmp
```

Ridge: RMSE

In [47]:
```
min_ridge_rmse
```

Out[47]:
```
1.014233447104668
```

Lasso (retreino)

In [48]:
```
model_lasso = LassoCV(alphas = [1, 0.75, 0.5, 0.25, 0.1, 0.075, 0.05, 0.025, 0.01,
```

Lasso: RMSE

In [49]:
```
rmse_cv(model_lasso, X_train, y).mean()
```

Out[49]:
```
1.0176304272725223
```

RandomForestRegressor (retreino)

In [50]:
```
y_test = test.votos

model_rf = RandomForestRegressor(n_estimators = 150, random_state = 42, max_depth=

# Train the model on training data
model_rf.fit(X_train, y);
```

RandomForestRegressor: RMSE

In [51]:
```
rmse_cv(model_rf, X_train, y).mean()
```

Out[51]:
```
0.9879650357238319
```

4.1) Responder: Dentre os modelos avaliados, qual foi o que deu o melhor resultado nos dados de 2014 em termos de RMSE? Justifique bem sua resposta.

Dentre os modelos avaliados, o Ridge obteve um RMSE de 1.014233447104668, enquanto o Lasso obteve um RMSE de 1.0176304272725223 e o Random Forest de 0.9879650357238319. Desta forma, o modelo que apresentou o melhor resultado para os dados de 2014 foi o modelo Random Forest. O RMSE (root mean squared error) é a medida que calcula a raiz quadrática média dos erros entre os valores observados e predições. Desta forma, quanto menor o RMSE melhor o modelo.

In [ ]:
```

```