In [159]:

```
# imports
import pandas as pd
import matplotlib.pyplot as plt
# follow the usual sklearn pattern: import, instantiate, fit
from sklearn.linear_model import LinearRegression
import numpy as np

# this allows plots to appear directly in the notebook
%matplotlib inline

# read data into a DataFrame
data = pd.read_csv('SF Listings - Ex Zillow.csv', index_col=0)
del data['Term']
del data['URL']
del data['Source']
del data['Rented']
data['Date'] = pd.to_datetime(data['Date'])

# create year dummy variables (because date isn't very intuitive variable)
data["Year"] = pd.DatetimeIndex(data["Date"]).to_period('Y')

# create dummy variables using get_dummies, then exclude the first dummy c
year_dummies = pd.get_dummies(data.Year, prefix='Year').iloc[:, :-1]

# print out baseline neighborhood
base_area = pd.get_dummies(data.Neighborhood, prefix='Neighborhood').iloc[
print('Base neighborhood: %s' % base_area)

# create dummy variables using get_dummies, then exclude the first dummy c
area_dummies = pd.get_dummies(data.Neighborhood, prefix='Neighborhood').il

# concatenate the dummy variable columns onto the original DataFrame (axis
data = pd.concat([data, area_dummies, year_dummies], axis=1)

data.head()
```

```
Base neighborhood: Neighborhood_Alamo Square
```

Out[159]:

	Neighborhood	Bedrooms	Bathrooms	Price	Sqft	Date	Year	Neighborhood_Ba
Address								
539 Octavia Street #9	Hayes Valley	0	1	1500	180	2015-03-18	2015	0

539 Octavia Street #11	Hayes Valley	0	1	1600	200	2015-03-30	2015	0	
539 Octavia Street #14	Hayes Valley	0	1	1850	221	2015-05-14	2015	0	
539 Octavia Street #12	Hayes Valley	0	1	1800	240	2015-04-16	2015	0	
539 Octavia Street #13	Hayes Valley	0	1	1995	280	2015-02-01	2015	0	

5 rows × 65 columns

In [160]:

```
# filter out any outliers, defined as rent >$10k or >2,500 sq ft

data = data[(data.Sqft <= 2500) & (data.Price <= 8000) & (data.Bedrooms <=
```

In [161]:

```
# FACTORING BY YEAR AND NEIGHBORHOOD
# Thesis: Neighborhoods influence valuations as a multiplier, rather than
# a square foot in SOMA is worth more than a square foot in Portrero by X%
# New model will look like this:
#      Price = B_1 x (SOMA Coeff * Year Coeff * Sqft) + intercept
#      $3,900 = B_1 x (1.20% * 1.15% * 2,023 sqft) + intercept
# where B_1 represents the price per square foot in base year and base nei
# I will ignore intercepts for now FIXME
# calculate the coefficients for the following matrix and save them for la
#                    SOMA    Mission    Portrero    Intercept
#  Price/SQFT      $1.23    $0.59       $0.88        $_.__

# create Price per square foot

price_per_foot = data.Price / data.Sqft
price_per_foot.name = 'price_per_foot'
data = pd.concat([data, price_per_foot], axis=1)

data.head()
```

Out[161]:

	Neighborhood	Bedrooms	Bathrooms	Price	Sqft	Date	Year	Neighborhood_Ba
Address								
539 Octavia Street #9	Hayes Valley	0	1	1500	180	2015-03-18	2015	0
539 Octavia Street #11	Hayes Valley	0	1	1600	200	2015-03-30	2015	0
539 Octavia Street #14	Hayes Valley	0	1	1850	221	2015-05-14	2015	0
539 Octavia Street #12	Hayes Valley	0	1	1800	240	2015-04-16	2015	0
539 Octavia Street #13	Hayes Valley	0	1	1995	280	2015-02-01	2015	0

5 rows × 66 columns

In [162]:
```
feature_cols = area_dummies.columns

X = data[feature_cols]
y = data.price_per_foot

# instantiate, fit
lm = LinearRegression()
lm.fit(X, y)

# print coefficients
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((lm.predict(X) - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % lm.score(X, y))

# print raw results
```

```
# print raw results
print("Intercept: %.2f" % lm.intercept_)

zip(feature_cols,lm.coef_)
```

```
Residual sum of squares: 0.60
Variance score: 0.51
3.28947368421
```

Out[162]:
```
[('Neighborhood_Bayview', 0.22919224160121199),
 ('Neighborhood_Berkeley', -0.46804511278195582),
 ('Neighborhood_Bernal Heights', 0.16135688057351771),
 ('Neighborhood_Buena Vista', 0.40915645277574719),
 ('Neighborhood_Candlestick Point', -0.367436740665263),
 ('Neighborhood_Central Richmond', -0.42195231668915678),
 ('Neighborhood_Central Waterfront', 0.88952843060745468),
 ('Neighborhood_Clarendon Heights', -2.2624743754446149e-15),
 ('Neighborhood_Cole Valley-Parnassus Heights', 1.0182186234817743),
 ('Neighborhood_Diamond Heights', -0.48096304591265182),
 ('Neighborhood_Dogpatch', -0.031787007783011578),
 ('Neighborhood_Downtown', 1.1549707602339019),
 ('Neighborhood_Downtown Oakland', -1.3728070175438778),
 ('Neighborhood_Downtown San Francisco', -0.23708338586672215),
 ('Neighborhood_Duboce Triangle', 1.2105263157894761),
 ('Neighborhood_Emeryville', -0.59662958948507128),
 ('Neighborhood_Eureka Valley', 0.85338345864660869),
 ('Neighborhood_Eureka Valley-Dolores Heights', 0.97538183064498296),
 ('Neighborhood_Excelsior', -0.06725146198830606),
 ('Neighborhood_Financial District', 1.078342407743498),
 ('Neighborhood_Glen Park', -0.56220095693779359),
 ('Neighborhood_Golden Gate Heights', 0.54385964912279716),
 ('Neighborhood_Hayes Valley', 3.1052765373898197),
 ('Neighborhood_Ingleside', -0.21793402635672438),
 ('Neighborhood_Inner Mission', 1.3417127947930085),
 ('Neighborhood_Lone Mountain', 0.1410025062656548),
 ('Neighborhood_Lower Pacific Heights', 1.0722288619764844),
 ('Neighborhood_Marina', 0.98156035635296712),
 ('Neighborhood_Mission Bay', 0.77019003759503779),
 ('Neighborhood_Mission Dolores', 2.0021929824561417),
 ('Neighborhood_Nob Hill', 1.4480367503101199),
 ('Neighborhood_Noe Valley', 0.38617181561812064),
 ('Neighborhood_North Beach', 0.28691520467835996),
 ('Neighborhood_North Panhandle', 0.75598086124401465),
 ('Neighborhood_North Waterfront', 1.9516164625400068),
 ('Neighborhood_Oakland', -0.56059084657921054),
 ('Neighborhood_Outer Richmond', 0.46052631578947029),
 ('Neighborhood_Pacific Heights', 0.62328082723319056),
 ('Neighborhood_Pacifica', -0.87255525520751076),
 ('Neighborhood_Portola', -0.55072227813629104),
 ('Neighborhood_Potrero Hill', 0.86222161507173978),
 ('Neighborhood_Rincon Hill', 1.244545438832916),
```

```
('Neighborhood_Russian Hill', -3.4503844708451225e-16),
('Neighborhood_SOMA', 0.64538128999443811),
('Neighborhood_South Beach', 1.079712752291329),
('Neighborhood_South Financial District', 1.7105263157894799),
('Neighborhood_Telegraph Hill', 1.5215983720284867),
('Neighborhood_Van Ness-Civic Center', 0.28866855912263772),
('Neighborhood_Visitacion Valley', -1.351271437019522),
('Neighborhood_Walnut Creek', -0.67671539284050231),
('Neighborhood_West Oakland', -0.81157530447165893),
('Neighborhood_Western Addition', 0.63933018196870939),
('Neighborhood_Westwood Park', -0.86977347007776606),
('Neighborhood_Yerba Buena', 0.93940092795471952)]
```

In [163]:
```
full_price = [lm.intercept_] * len(lm.coef_)
full_price += lm.coef_

area_price_per_foot = dict(zip(feature_cols,full_price))
area_price_per_foot[base_area] = lm.intercept_

print area_price_per_foot


# calculate the coefficients for the following matrix:
#              2011    2012    2013    2014    2015
#  SQFT
```

```
{'Neighborhood_Buena Vista': 3.6986301369862762, 'Neighborhood_Nob Hill':
ghborhood_Lower Pacific Heights': 4.3617025461870131, 'Neighborhood_Claren
4210527, 'Neighborhood_Alamo Square': 3.2894736842105292, 'Neighborhood_Go
3333333333264, 'Neighborhood_Outer Richmond': 3.7499999999999996, 'Neighb
3.4508305647840469, 'Neighborhood_Westwood Park': 2.4197002141327633, 'Nei
4.5340191230428211, 'Neighborhood_Oakland': 2.7288828376313186, 'Neighborh
4736842105288, 'Neighborhood_South Beach': 4.3691864365018578, 'Neighborho
8571428571379, 'Neighborhood_Financial District': 4.367816091954027, 'Neig
ancisco': 3.0523902983438069, 'Neighborhood_North Beach': 3.576388888888888
l Richmond': 2.8675213675213724, 'Neighborhood_Mission Dolores': 5.2916666
_Excelsior': 3.2222222222222232, 'Neighborhood_Visitacion Valley': 1.93820
od_Duboce Triangle': 4.5000000000000053, 'Neighborhood_Lone Mountain': 3.4
orhood_Hayes Valley': 6.3947502216003489, 'Neighborhood_Cole Valley-Parnas
7692303, 'Neighborhood_Dogpatch': 3.2576866764275176, 'Neighborhood_North
545441, 'Neighborhood_Pacific Heights': 3.9127545114437199, 'Neighborhood_
8038661792388, 'Neighborhood_Yerba Buena': 4.2288746121652485, 'Neighborho
272727355, 'Neighborhood_Noe Valley': 3.6756454998286499, 'Neighborhood_In
8048, 'Neighborhood_Portola': 2.7387514060742379, 'Neighborhood_West Oakla
'Neighborhood_Eureka Valley-Dolores Heights': 4.2648555148555118, 'Neighbo
strict': 5.0000000000000089, 'Neighborhood_Bayview': 3.5186659258117414, '
k': 2.6127582913700271, 'Neighborhood_Mission Bay': 4.0596637218055669, 'N
.4444444444444313, 'Neighborhood_North Waterfront': 5.241090146750536, 'Ne
l': 4.8110720562390163, 'Neighborhood_Pacifica': 2.4169184290030183, 'Neig
14285714285734, 'Neighborhood_Potrero Hill': 4.1516952992822693, 'Neighbor
1.9166666666666514, 'Neighborhood_Central Waterfront': 4.1790021148179841,
ck Point': 2.9220369435452662, 'Neighborhood_Inner Mission': 4.63118647900
ryville': 2.6928440947254577, 'Neighborhood_Marina': 4.2710340405634959, '
ghts': 2.8085106382978773, 'Neighborhood_SOMA': 3.9348549742049674, 'Neigh
enter': 3.5781422433331671}
```

In [164]:
```
# calculate the multipliers for each neighborhood relative to base area
# SOMA_mult = SOMA_per_foot / Base_per_foot

area_mults = [lm.intercept_] * len(lm.coef_)
area_mults = full_price / area_mults - [1]*len(lm.coef_)
```

```
zip(feature_cols, area_mults)
```

```
Out[164]: [('Neighborhood_Bayview', 0.069674441446768487),
 ('Neighborhood_Berkeley', -0.14228571428571446),
 ('Neighborhood_Bernal Heights', 0.049052491694349332),
 ('Neighborhood_Buena Vista', 0.12438356164382691),
 ('Neighborhood_Candlestick Point', -0.1117007691622398l),
 ('Neighborhood_Central Richmond', -0.12827350427350359),
 ('Neighborhood_Central Waterfront', 0.27041664290466616),
 ('Neighborhood_Clarendon Heights', -6.6613381477509392e-16),
 ('Neighborhood_Cole Valley-Parnassus Heights', 0.30953846153845888),
 ('Neighborhood_Diamond Heights', -0.14621276595744603),
 ('Neighborhood_Dogpatch', -0.0096632503660355473),
 ('Neighborhood_Downtown', 0.35111111111110582),
 ('Neighborhood_Downtown Oakland', -0.41733333333333844),
 ('Neighborhood_Downtown San Francisco', -0.072073349303483525),
 ('Neighborhood_Duboce Triangle', 0.36800000000000033),
 ('Neighborhood_Emeryville', -0.18137539520346158),
 ('Neighborhood_Eureka Valley', 0.2594285714285689),
 ('Neighborhood_Eureka Valley-Dolores Heights', 0.29651607651607437),
 ('Neighborhood_Excelsior', -0.020444444444445042),
 ('Neighborhood_Financial District', 0.32781609195402295),
 ('Neighborhood_Glen Park', -0.17090909090908912),
 ('Neighborhood_Golden Gate Heights', 0.16533333333333022),
 ('Neighborhood_Hayes Valley', 0.94400406736650444),
 ('Neighborhood_Ingleside', -0.066251944012444164),
 ('Neighborhood_Inner Mission', 0.40788068961707413),
 ('Neighborhood_Lone Mountain', 0.042864761904758852),
 ('Neighborhood_Lower Pacific Heights', 0.32595757404085091),
 ('Neighborhood_Marina', 0.29839434833130163),
 ('Neighborhood_Mission Bay', 0.23413777142889125),
 ('Neighborhood_Mission Dolores', 0.60866666666666669),
 ('Neighborhood_Nob Hill', 0.44020317209427606),
 ('Neighborhood_Noe Valley', 0.11739623194790849),
 ('Neighborhood_North Beach', 0.087222222222221424),
 ('Neighborhood_North Panhandle', 0.22981818181818037),
 ('Neighborhood_North Waterfront', 0.59329140461216157),
 ('Neighborhood_Oakland', -0.17041961736007993),
 ('Neighborhood_Outer Richmond', 0.13999999999999879),
 ('Neighborhood_Pacific Heights', 0.18947737147888977),
 ('Neighborhood_Pacifica', -0.26525679758308307),
 ('Neighborhood_Portola', -0.16741957255343243),
 ('Neighborhood_Potrero Hill', 0.2621153709818087),
 ('Neighborhood_Rincon Hill', 0.37834181340501649),
 ('Neighborhood_Russian Hill', -1.1102230246251565e-16),
 ('Neighborhood_SOMA', 0.196195912158309),
 ('Neighborhood_South Beach', 0.32823267669656353),
 ('Neighborhood_South Financial District', 0.52000000000000135),
```

```
 ('Neighborhood_Telegraph Hill', 0.46256590509665974),
 ('Neighborhood_Van Ness-Civic Center', 0.087755241973281883),
 ('Neighborhood_Visitacion Valley', -0.41078651685393443),
 ('Neighborhood_Walnut Creek', -0.20572147942351249),
 ('Neighborhood_West Oakland', -0.24671889255938406),
 ('Neighborhood_Western Addition', 0.1943563753184876),
 ('Neighborhood_Westwood Park', -0.26441113490364065),
 ('Neighborhood_Yerba Buena', 0.28557788209823443)]
```

In [165]:

```
# calculate the adjusted Sqft (Sqft * Area_mult) for the dataset and add i

# for each property, multiplier is sum of array [area_dummies] x [area_mul

t = data[area_dummies.columns] * area_mults
t = t.T.sum()

t.name = 'area_multiplier'
t = t + 1
data = pd.concat([data, t], axis=1)

adj_sqft = data.Sqft * t
adj_sqft.name = 'area_adj_sqft'
data = pd.concat([data, adj_sqft], axis=1)

data.head()
```

Out[165]:

	Neighborhood	Bedrooms	Bathrooms	Price	Sqft	Date	Year	Neighborhood_Ba
Address								
539 Octavia Street #9	Hayes Valley	0	1	1500	180	2015-03-18	2015	0
539 Octavia Street #11	Hayes Valley	0	1	1600	200	2015-03-30	2015	0
539 Octavia Street #14	Hayes Valley	0	1	1850	221	2015-05-14	2015	0
539								

Octavia Street #12	Hayes Valley	0	1	1800	240	2015-04-16	2015	0
539 Octavia Street #13	Hayes Valley	0	1	1995	280	2015-02-01	2015	0

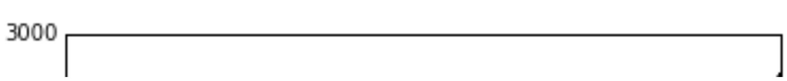
5 rows × 68 columns

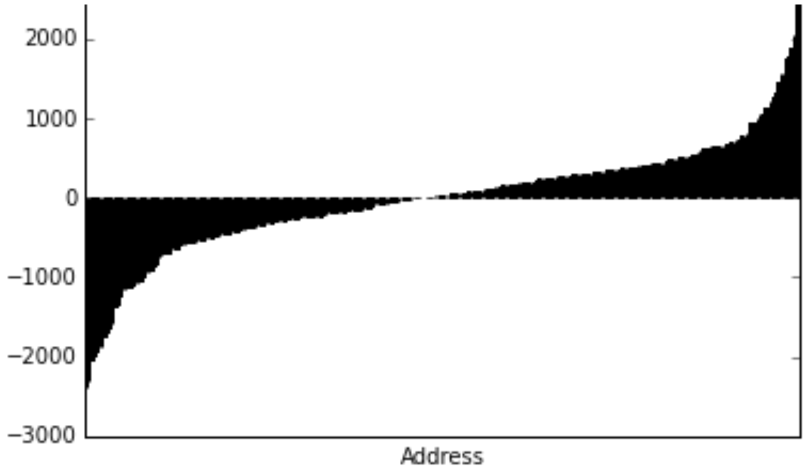
In [187]:

```
# run the regression based on area_adj_sqft rather than sqft

# create X and y
feature_cols = [data.area_adj_sqft.name]

X = data[feature_cols]
y = data.Price

# instantiate, fit
lm = LinearRegression()
lm.fit(X, y)

# print coefficients
print("Intercept: %.2f" % lm.intercept_)

# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((lm.predict(X) - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % lm.score(X, y))
zip(feature_cols, lm.coef_)

# calculate predictions for the data set and plot errors
predictions = lm.predict(X)
errors = predictions-y
errors.name = 'Error'

# visualize the relationship between the features and the response using s
errors.sort()
errors.plot(kind='bar').get_xaxis().set_ticks([])
```

```
1142.54962838
Residual sum of squares: 507164.94
Variance score: 0.73
```

Out[187]: []





In [183]:
```
feature_cols = year_dummies.columns

X = data[feature_cols]
y = data.price_per_foot

# instantiate, fit
lm = LinearRegression()
lm.fit(X, y)

# print coefficients
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((lm.predict(X) - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % lm.score(X, y))

# print raw results
print lm.intercept_

zip(feature_cols,lm.coef_)
```

```
Residual sum of squares: 1.16
Variance score: 0.06
4.43106702515
```

Out[183]:
```
[(u'Year_2011', -0.26050619124350716),
 (u'Year_2012', -1.0047722044318308),
 (u'Year_2013', -0.68048935241674591),
 (u'Year_2014', -0.47727887796110385)]
```

In [184]:
```
full_price = [lm.intercept_] * len(lm.coef_)
full_price += lm.coef_

year_price_per_foot = dict(zip(feature_cols,full_price))
year_price_per_foot[base_area] = lm.intercept_

print year_price_per_foot
```

```
{u'Year_2012': 3.4262948207171355, u'Year_2013': 3.75057767273222, 'Neighb
4310670251489661, u'Year_2011': 4.1705608339054585, u'Year_2014': 3.953788
```

In [185]:
```
# calculate the multipliers for each year relative to base year
# 2014_mult = 2014_per_foot / 2015_per_foot

year_mults = [lm.intercept_] * len(lm.coef_)
year_mults = full_price / year_mults - [1]*len(lm.coef_)

zip(feature_cols, year_mults)
```

Out[185]:
```
[(u'Year_2011', -0.058790848742521495),
 (u'Year_2012', -0.22675626406216498),
 (u'Year_2013', -0.15357234466428971),
 (u'Year_2014', -0.10771195182836546)]
```

In [186]:
```
# calculate the adjusted Sqft (Sqft * Year_mult) for the dataset and add i

# for each property, multiplier is sum of array [year_dummies] x [year_mul

t = data[year_dummies.columns] * year_mults
t = t.T.sum()

t.name = 'year_multiplier'
t = t + 1
data = pd.concat([data, t], axis=1)

year_adj_sqft = data.area_adj_sqft * t
year_adj_sqft.name = 'year_and_area_adj_sqft'
data = pd.concat([data, year_adj_sqft], axis=1)

data.head()
```

Out[186]:

	Neighborhood	Bedrooms	Bathrooms	Price	Sqft	Date	Year	Neighborhood_Ba
Address								
539 Octavia Street	Hayes Valley	0	1	1500	180	2015-03-18	2015	0

#9								
539 Octavia Street #11	Hayes Valley	0	1	1600	200	2015-03-30	2015	0
539 Octavia Street #14	Hayes Valley	0	1	1850	221	2015-05-14	2015	0
539 Octavia Street #12	Hayes Valley	0	1	1800	240	2015-04-16	2015	0
539 Octavia Street #13	Hayes Valley	0	1	1995	280	2015-02-01	2015	0

5 rows × 70 columns

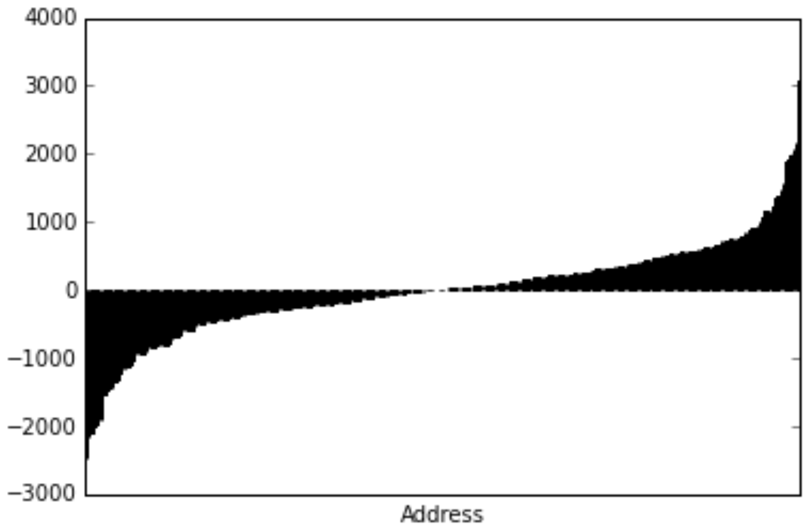
In [188]:

```
# run the regression based on year_and_area_adj_sqft rather than area_adj_

# create X and y
feature_cols = [data.year_and_area_adj_sqft.name]

X = data[feature_cols]
y = data.Price

# instantiate, fit
lm = LinearRegression()
lm.fit(X, y)

# print coefficients
print lm.intercept_
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((lm.predict(X) - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % lm.score(X, y))
zip(feature_cols, lm.coef_)

# calculate predictions for the data set and plot errors
predictions = lm.predict(X)
errors = predictions-y
errors.name = 'Error'
```

```
# visualize the relationship between the features and the response using s
errors.sort()
errors.plot(kind='bar').get_xaxis().set_ticks([])
```

```
1115.70341642
Residual sum of squares: 507813.90
Variance score: 0.73
```

Out[188]: []



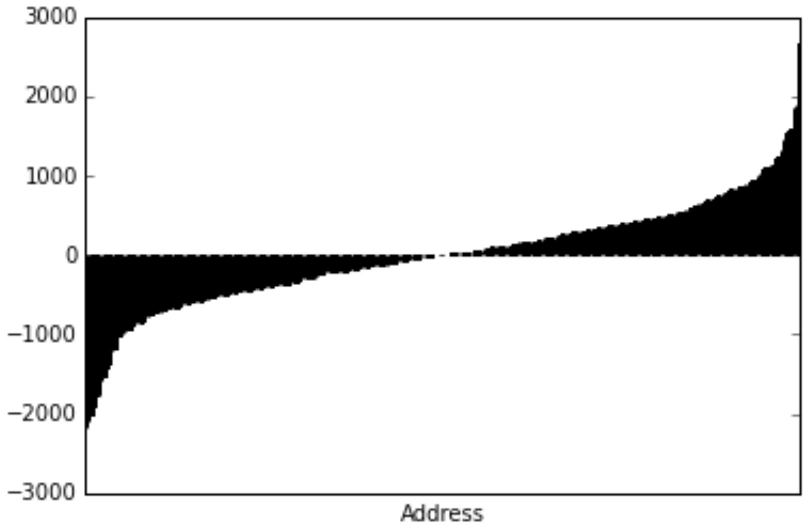
In [192]:

```
# add back bedrooms and bathrooms to the regression, technically should be

# create X and y
feature_cols = [data.year_and_area_adj_sqft.name, 'Bedrooms','Bathrooms']

X = data[feature_cols]
y = data.Price

# instantiate, fit
lm = LinearRegression()
lm.fit(X, y)

# print coefficients
print("Intercept: %.2f" % lm.intercept_)
# The mean square error
print("Residual sum of squares: %.2f"
      % np.mean((lm.predict(X) - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % lm.score(X, y))
print zip(feature_cols, lm.coef_)

# calculate predictions for the data set and plot errors
predictions = lm.predict(X)
```

```
errors = predictions-y
errors.name = 'Error'

# visualize the relationship between the features and the response using s
errors.sort()
errors.plot(kind='bar').get_xaxis().set_ticks([])
```

```
Intercept: 876.19
Residual sum of squares: 450969.05
Variance score: 0.76
[('year_and_area_adj_sqft', 2.055160552581313), ('Bedrooms', 198.664264091
0.54413127330679)]
```

Out[192]: []

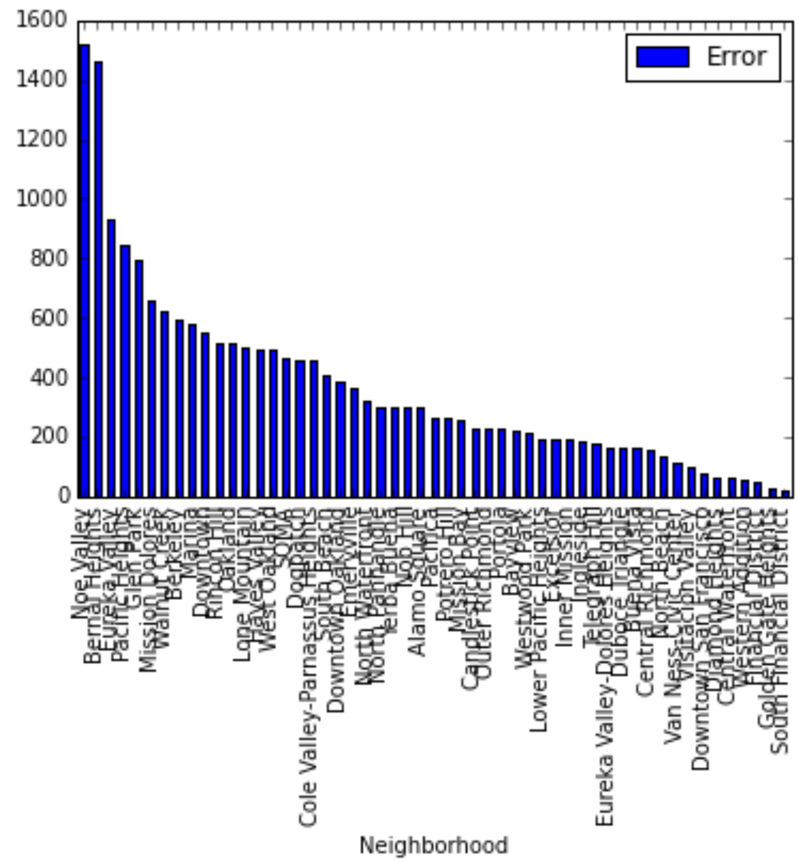


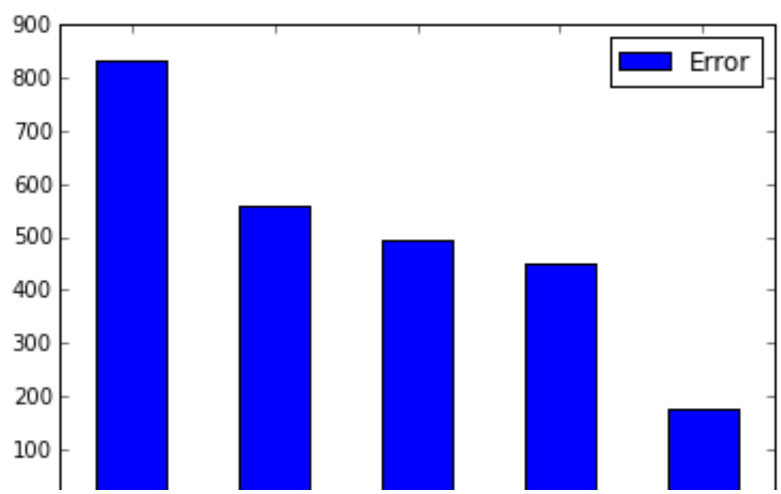
In [189]:

```
# show errors by neighborhood to see if there are any neighborhoods with f

hooderrors = data[['Neighborhood']]

errors = predictions-y
errors.name = 'Error'

hooderrors = pd.concat([hooderrors,errors.abs()],axis=1)

hood_group = hooderrors.groupby('Neighborhood')

import numpy
def median(lst):
    return numpy.median(numpy.array(lst))

error_avg = hood_group.median()
error_avg.sort(columns='Error',ascending=False).plot(kind='bar')

# show errors by year to see if there are any years with funky differences
```

```
yearerrors = data[['Year']]

yearerrors = pd.concat([yearerrors,errors.abs()],axis=1)

year_group = yearerrors.groupby('Year')
error_avg = year_group.mean()
error_avg.sort(columns='Error',ascending=False).plot(kind='bar')
```

Out[189]: <matplotlib.axes._subplots.AxesSubplot at 0x115ef1c50>





In [ ]:
```
class ListTable(list):
    """ Overridden list class which takes a 2-dimensional list of
        the form [[1,2,3],[4,5,6]], and renders an HTML Table in
        IPython Notebook. """

    def _repr_html_(self):
        html = ["<table>"]
        for row in self:
            html.append("<tr>")

            for col in row:
                html.append("<td>{0}</td>".format(col))

            html.append("</tr>")
        html.append("</table>")
        return ''.join(html)

table = ListTable()

dtype = [('Effect', 'S100'), ('Coefficient', float)]

# round to pennies
round_coef = map(round,lm.coef_,[2]*len(lm.coef_))
x = np.array(zip(feature_cols, round_coef),dtype=dtype)
x.T
x = np.sort(x,axis=0,order='Coefficient')

table.append(['Effect','Coefficient'])
for i in x:
    table.append(i)

print "Intercept: $"+`round(lm.intercept_)`
table
```

In [ ]:

In [ ]: