In [1]:
```
#
# Final Project for Cogs 109 Spring 2020.
# Dataset: new-york-city-airbnb-open-data
# Dataset link: https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data#
# Algorithm used: Linear regression, K-mean clustering
#
```

In [70]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import math
%matplotlib inline
```

In [71]:
```
# Hypothesis: the price of each airbnb listing is direct related to its number of
```

In [72]:
```
# Linear regression approach:
# Idea: First we use two models to see what the data looks like and how well each
# then we will use cross validation on the test data by taking the weight vector

# Load csv file into our data
df = pd.read_csv("AB_NYC_2019.csv")
# Check data size
print("Data size: ", len(df))
# Check data
df
```

```
Data size:  48895
```

In [73]:
```
# Extract the different room_types:
room_types = []
for type in df['room_type']:
    if type not in room_types:
        room_types.append(type)
print(room_types)
```

```
['Private room', 'Entire home/apt', 'Shared room']
```

In [74]:
```
# Here we can clearly see that there are 3 different room types:
#
```

```
# Models for experiement
# M1: price = w0 + w1 x number_of_reviews
# M2: price = w0 + w1 x number_of_reviews + w2 x availability_365^2
```

In [75]:
```
# Split data into training set and test set.
# Training set (30000 samples)
# Test set(18895 samples)

# First we trim the data down and extract the variables
Y = df['price'].values
X1 = df['number_of_reviews'].values
X2 = df['availability_365'].values

# First 30000 samples
X1_train = X1[:30000]
X2_train = X2[:30000]
Y_train = Y[:30000]
# Last 18895 samples
X1_test = X1[30000:]
X2_test = X2[:30000]
Y_test = Y[30000:]
# Check sizes
print(len(X1_train))
print(len(X2_train))
print(len(Y_train))
print(len(X1_test))
print(len(X2_test))
print(len(Y_test))
```

```
30000
30000
30000
18895
30000
18895
```

In [76]:
```
# Training data on model 1
ones = np.ones(len(X1_train),dtype=int).reshape(len(X1_train),1)
A1 = np.append(ones,X1_train.reshape(len(X1_train),1),axis=1)
# Calculate weight vector
w1 = np.linalg.lstsq(A1, Y_train,rcond=None)[0]
print("Model 1:")
print("Weight = ", w1)
```

```
Model 1:
Weight =  [155.06946056  -0.19731569]
```

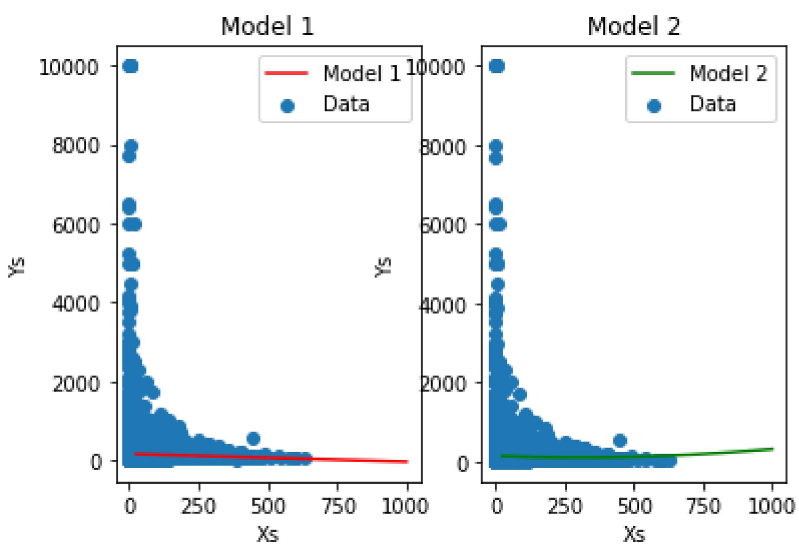
In [77]:
```
# Training data on model 2
ones = np.ones(len(X1_train))
squares = np.square(X2_train)

A2 = np.vstack([ones,X1_train,squares])
A2 = A2.T
print(A2.shape)

## Solve for w, the weight vector
w2 = np.linalg.lstsq(A2, Y_train, rcond=None)[0]

print("Model 2:")
print("Weight = ", w2)
```

```
(30000, 3)
Model 2:
Weight =  [ 1.45584887e+02 -2.66527187e-01  4.38539180e-04]
```

In [78]:
```
# Plot the two models
## Create a smooth set of X values for plotting the model
lineinput = np.linspace(25, 1000, 30000)
## Send the X values for plotting through the linear model
ones = np.ones(len(lineinput))
squares = np.square(lineinput)

A1 = np.vstack([ones,lineinput])
A2 = np.vstack([ones,lineinput,squares])
yplot1 = np.matmul(A1.T, w1)
yplot2 = np.matmul(A2.T, w2)
```

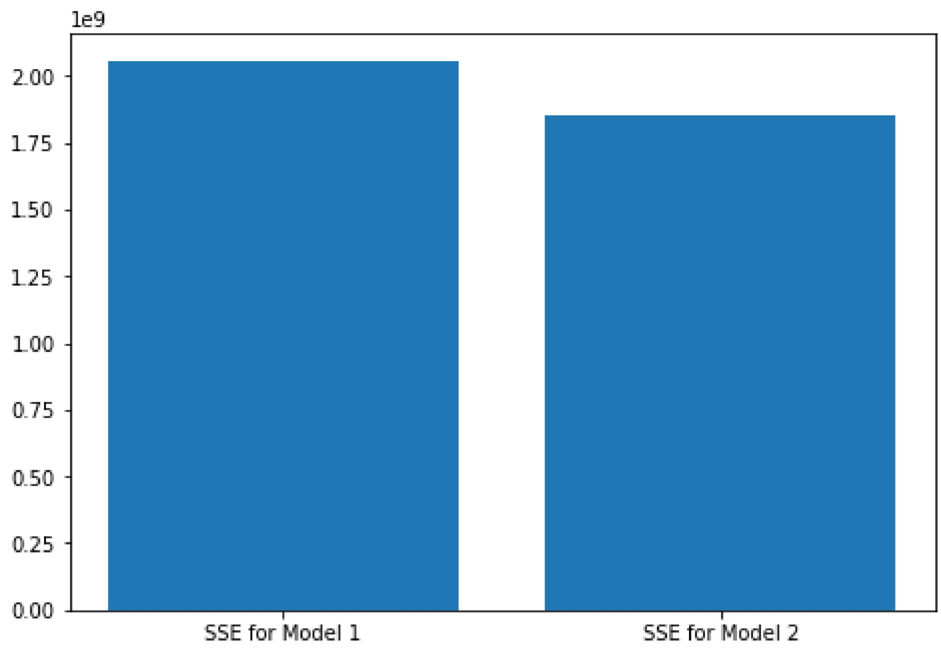
In [79]:
```
## Plot the data along with the models
fig_a, ax  = plt.subplots(1,2)
ax[0].scatter(X1_train,Y_train,label='Data')
ax[0].plot(lineinput, yplot1,color='r',label='Model 1')
ax[0].set_title('Model 1')
ax[0].set_xlabel('Xs')
ax[0].set_ylabel('Ys')
ax[0].legend()

ax[1].scatter(X1_train,Y_train,label='Data')
ax[1].plot(lineinput, yplot2,color='g',label='Model 2')
ax[1].set_title('Model 2')
ax[1].set_xlabel('Xs')
ax[1].set_ylabel('Ys')
ax[1].legend()
plt.show
```

Out[79]: <function matplotlib.pyplot.show(*args, **kw)>



In [80]:
```
# Calculating SSEs
SSE_1 = SSE_2 = 0

for i in range(len(Y_train)):
    SSE_1 += (yplot1[i] - Y_train[i])**2
    SSE_2 += (yplot2[i] - Y_train[i])**2

print("SSE for Model 1: ", SSE_1)
print("SSE for Model 2: ", SSE_2)

# Bar plot
# x = range(2)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['SSE for Model 1','SSE for Model 2']
SSEs = [SSE_1, SSE_2]
ax.bar(langs, SSEs)
plt.show()
```

```
SSE for Model 1:  2059027191.8865726
SSE for Model 2:  1852165084.3972049
```



In [81]:
```
# As we can see from the above curve and histogram of the SSE, model 2 has slight
# points.
# But before we move on to the test data, let us use redo the plots and this time
# Model 3: price = w0 + w1 x availability_365 + w2 x number_of_reviews^2
# This time we are putting more weight on number of reviews,
```

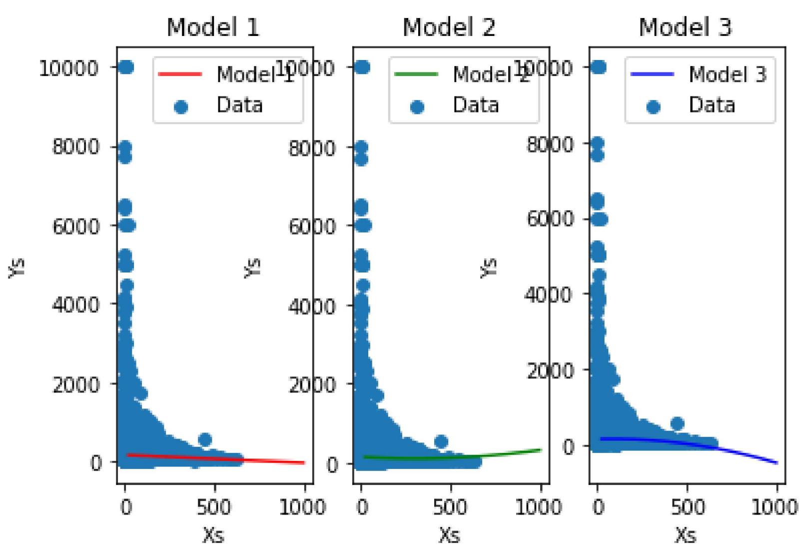
In [90]:
```
# First 30000 samples
X1_train = X2[:30000]
X2_train = X1[:30000]
Y_train = Y[:30000]

# Training data on model 3
ones = np.ones(len(X1_train))
squares = np.square(X2_train)

A3 = np.vstack([ones,X1_train,squares])
A3 = A3.T
print(A2.shape)

## Solve for w, the weight vector
w3 = np.linalg.lstsq(A3, Y_train, rcond=None)[0]

print("Model 3:")
print("Weight = ", w3)
```

```
(3, 30000)
Model 3:
Weight =  [ 1.38847228e+02  1.29562827e-01 -7.54716118e-04]
```

In [91]:
```
## Create a smooth set of X values for plotting the model
lineinput = np.linspace(25, 1000, 30000)
## Send the X values for plotting through the linear model
ones = np.ones(len(lineinput))
squares = np.square(lineinput)

# A1 = np.vstack([ones,lineinput])
A3 = np.vstack([ones,lineinput,squares])
# yplot1 = np.matmul(A1.T, w1)
yplot3 = np.matmul(A3.T, w3)
```

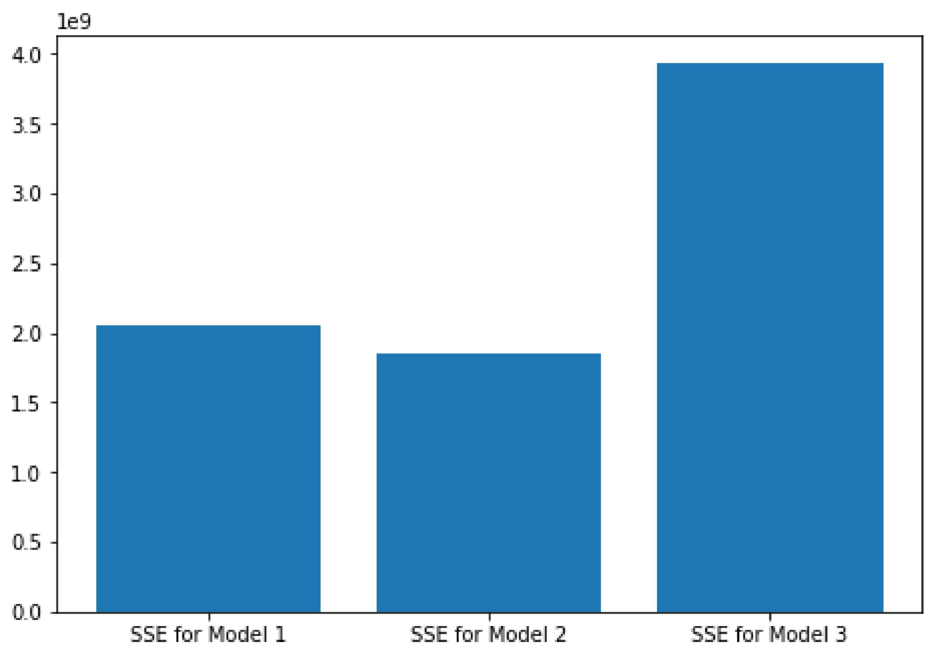
In [92]:
```
## Plot the data along with the model
fig_a, ax  = plt.subplots(1,3)
ax[0].scatter(X2_train,Y_train,label='Data')
ax[0].plot(lineinput, yplot1,color='r',label='Model 1')
ax[0].set_title('Model 1')
ax[0].set_xlabel('Xs')
ax[0].set_ylabel('Ys')
ax[0].legend()

ax[1].scatter(X2_train,Y_train,label='Data')
ax[1].plot(lineinput, yplot2,color='g',label='Model 2')
ax[1].set_title('Model 2')
ax[1].set_xlabel('Xs')
ax[1].set_ylabel('Ys')
ax[1].legend()

ax[2].scatter(X2_train,Y_train,label='Data')
ax[2].plot(lineinput, yplot3,color='b',label='Model 3')
ax[2].set_title('Model 3')
ax[2].set_xlabel('Xs')
ax[2].set_ylabel('Ys')
ax[2].legend()
plt.show
```

Out[92]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [93]:
```
# Calculating SSEs
SSE_1 = SSE_2 = SSE_3 = 0

for i in range(len(Y_train)):
    SSE_1 += (yplot1[i] - Y_train[i])**2
    SSE_2 += (yplot2[i] - Y_train[i])**2
    SSE_3 += (yplot3[i] - Y_train[i])**2

print("SSE for Model 1: ", SSE_1)
print("SSE for Model 2: ", SSE_2)
print("SSE for Model 2: ", SSE_3)

# Bar plot
# x = range(2)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['SSE for Model 1','SSE for Model 2', 'SSE for Model 3']
SSEs = [SSE_1, SSE_2, SSE_3]
ax.bar(langs, SSEs)
plt.show()
```

```
SSE for Model 1:  2059027191.8865726
SSE for Model 2:  1852165084.3972049
SSE for Model 2:  3941829602.1140156
```



In [94]:
```
# Now to my surprise, the third model which puts more emphasis on the number of r
# SSE.
# Now we will train the data on our final model.
# Model 4: price = w0 + w1 x number_of_reviews + w2 x calculated_host_listings_co
```

In [95]:
```
X3 = df['calculated_host_listings_count'].values
X3_train = X3[:30000]

# Training data on model 4
ones = np.ones(len(X1_train))
squares = np.square(X3_train)

A4 = np.vstack([ones,X1_train,squares])
A4 = A4.T
print(A2.shape)

## Solve for w, the weight vector
w4 = np.linalg.lstsq(A4, Y_train, rcond=None)[0]

print("Model 4:")
print("Weight = ", w4)
```

```
(3, 30000)
Model 4:
Weight =  [1.37348729e+02 1.07136266e-01 4.89861107e-03]
```

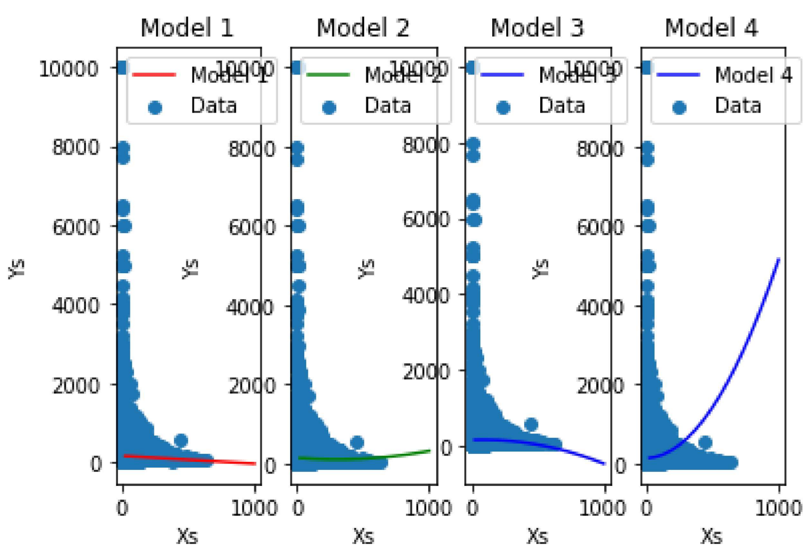
In [96]:
```
## Create a smooth set of X values for plotting the model
lineinput = np.linspace(25, 1000, 30000)
## Send the X values for plotting through the linear model
ones = np.ones(len(lineinput))
squares = np.square(lineinput)

# A1 = np.vstack([ones,lineinput])
A4 = np.vstack([ones,lineinput,squares])
# yplot1 = np.matmul(A1.T, w1)
yplot4 = np.matmul(A4.T, w4)
```

In [98]:

```python
## Plot the data along with the model
fig_a, ax  = plt.subplots(1,4)
ax[0].scatter(X2_train,Y_train,label='Data')
ax[0].plot(lineinput, yplot1,color='r',label='Model 1')
ax[0].set_title('Model 1')
ax[0].set_xlabel('Xs')
ax[0].set_ylabel('Ys')
ax[0].legend()

ax[1].scatter(X2_train,Y_train,label='Data')
ax[1].plot(lineinput, yplot2,color='g',label='Model 2')
ax[1].set_title('Model 2')
ax[1].set_xlabel('Xs')
ax[1].set_ylabel('Ys')
ax[1].legend()

ax[2].scatter(X2_train,Y_train,label='Data')
ax[2].plot(lineinput, yplot3,color='b',label='Model 3')
ax[2].set_title('Model 3')
ax[2].set_xlabel('Xs')
ax[2].set_ylabel('Ys')
ax[2].legend()

ax[3].scatter(X2_train,Y_train,label='Data')
ax[3].plot(lineinput, yplot4,color='b',label='Model 4')
ax[3].set_title('Model 4')
ax[3].set_xlabel('Xs')
ax[3].set_ylabel('Ys')
ax[3].legend()

plt.show
```

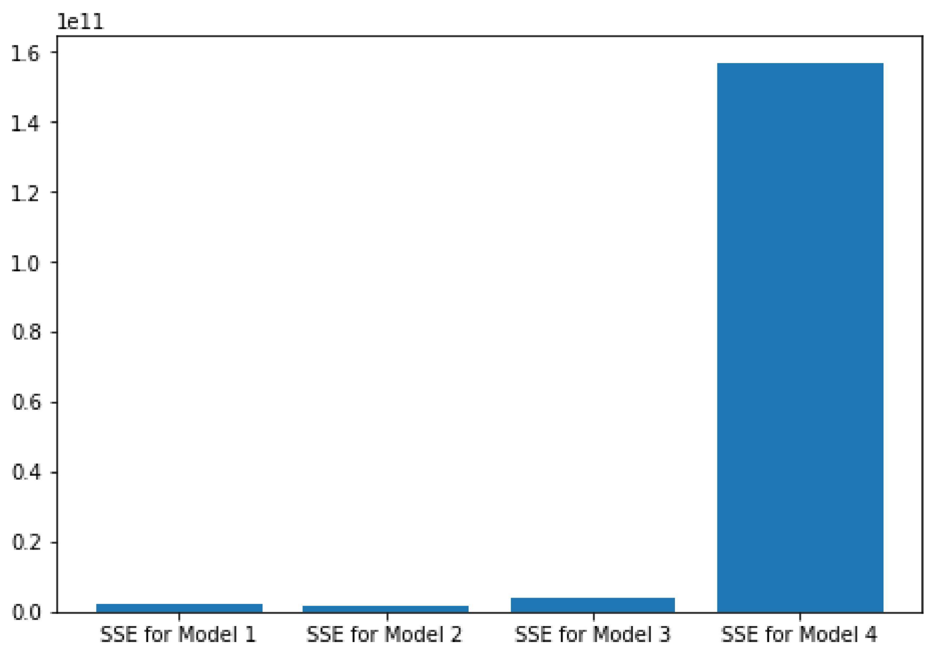
Out[98]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [99]:
```
# Calculating SSEs
SSE_1 = SSE_2 = SSE_3 = SSE_4 = 0

for i in range(len(Y_train)):
    SSE_1 += (yplot1[i] - Y_train[i])**2
    SSE_2 += (yplot2[i] - Y_train[i])**2
    SSE_3 += (yplot3[i] - Y_train[i])**2
    SSE_4 += (yplot4[i] - Y_train[i])**2

print("SSE for Model 1: ", SSE_1)
print("SSE for Model 2: ", SSE_2)
print("SSE for Model 2: ", SSE_3)
print("SSE for Model 2: ", SSE_4)

# Bar plot
# x = range(2)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['SSE for Model 1','SSE for Model 2', 'SSE for Model 3', 'SSE for Model 4
SSEs = [SSE_1, SSE_2, SSE_3, SSE_4]
ax.bar(langs, SSEs)
plt.show()
```

```
SSE for Model 1:  2059027191.8865726
SSE for Model 2:  1852165084.3972049
SSE for Model 2:  3941829602.1140156
SSE for Model 2:  157031517897.2471
```



In [111]:
```
# Model 4's SSE is just over the top, it fits the least amount of data points.
# Therefore, we will use Model 2 and run cross validation on our test data.
```

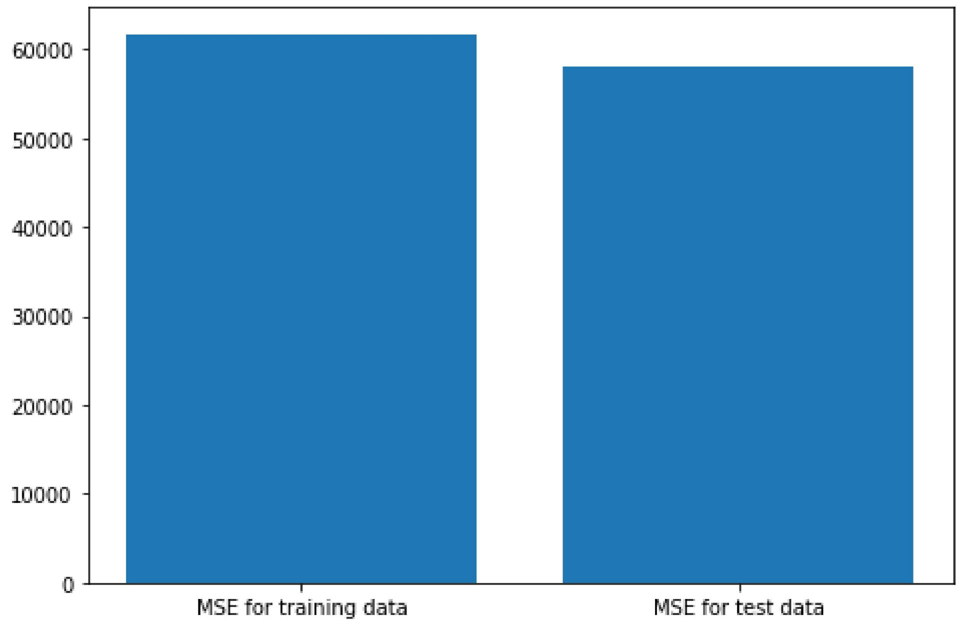
In [105]:
```
X = df[['number_of_reviews','availability_365']].values
X_test = X[30000:]
ones = np.ones(18895,dtype=int).reshape(18895,1)
A_test = np.append(ones,X_test.reshape(18895,2),axis=1)

Y_train_hat = np.matmul(A2.T, w2)
Y_test_hat = np.matmul(A_test, w2)
```

In [107]:
```
# Calculate MSEs
# Calculating SSEs
SSE_train = SSE_test = 0
MSE_train = MSE_test = 0

for i in range(len(Y_train)):
    SSE_train += (Y_train_hat[i] - Y_train[i])**2
MSE_train = SSE_train / len(X1_train)

for i in range(len(Y_test)):
    SSE_test += (Y_test_hat[i] - Y_test[i])**2
MSE_test = SSE_test / len(X_test)
```

In [108]:
```
print("MSE for training set: ", MSE_train)
print("MSE for test set: ", MSE_test)
```

```
MSE for training set:  61738.83614657349
MSE for test set:  57969.30518053395
```

In [109]:
```
# Bar plot
# x = range(2)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
langs = ['MSE for training data','MSE for test data']
MSEs = [MSE_train, MSE_test]
ax.bar(langs, MSEs)
plt.show()
```



In [110]:
```
# To my surpsise, the MSE for test data is actually lower than the MSE of the tra
# This shows that our model: price = w0 + w1 x number_of_reviews + w2 x availabil
# data set.
```

In [ ]:
```
```