MGMTMSA 403: Optimization

Assignment 3: Predicting Airbnb Prices

Due on BruinLearn by 11:59pm on February 7th.

Arnav Garg (906310841)
Oscar Trumpy (706310371)
Facundo Klappenbach (306316186)

Background

The file AirbnbTrain.csv contains data on 1700 Airbnb listings in Hollywood, CA. The dataset contains features such as the location (by latitude and longitude), number of guests accomodated, number of beds, and other variables. The dataset also contains the price per night of each Airbnb listing. Your assignment will be to formulate an optimization model to predict the price of Airbnb listings using this dataset.

Note: Given a set of model coefficients β1, β2, . . . , βd, the average prediction error of the model β for a data set (xi,yi), i = 1,...,n is given by

$$Error = \frac{1}{n}\sum_{i=1}^{n}|y_i - \sum_{j=1}^{d}\beta_j x_{ij}| \tag{1}$$

Questions

Model 1. Formulate the least absolute deviations regression problem as a linear program. Solve the linear program using the data given in the file AirbnbTrain.csv. What is the prediction error, in $/night, of your model on the test set (provided in AirbnbTest.csv)?

The solution can be given by the following equations:

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} Z_i \tag{2}$$

$$\text{s.t.} \quad Z_i \geq y_i - (\sum_{j=1}^{n} \beta_j \times X_{ij}) \quad i = 1 \dots m \tag{3}$$

$$Z_i \geq (\sum_{j=1}^{n} \beta_j \times X_{ij}) - y_i \quad i = 1 \dots m \tag{4}$$

In [ ]:
```
# Import libraries
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from gurobipy import *

# Load dataset
train = pd.read_csv('AirbnbTrain.csv')
test = pd.read_csv('AirbnbTest.csv')

# Display the first 5 rows of the dataset
train.head()
```

Out[ ]:

	latitude	longitude	Entire home	accommodates	bathrooms	bedrooms	beds	cleaning_fee	minimum_nights	number_of_reviews	review_sc
0	34.103701	-118.332241	1	13	2.0	3	2	150	2	1	
1	34.099484	-118.331645	1	8	2.0	2	4	150	1	11	
2	34.104321	-118.329662	1	4	1.0	0	1	55	1	1	
3	34.101028	-118.317848	0	2	1.0	1	1	20	1	8	
4	34.098292	-118.324980	1	2	1.0	1	1	20	1	11	

In [ ]:
```
y = train['price']
m = train.shape[0]
n = train.drop(columns=['price']).shape[1]
```

```
x = train.drop(columns=['price'])

################################################
# Initialize the model
mod = Model()

################################################
# Define Variables
z = mod.addVars(m)
b = mod.addVars(n)

################################################
# Define Constraints
for i in range(m):
    mod.addConstr(z[i] >= y[i] - sum(b[j] * x.iloc[i, j] for j in range(n - 1)))

for i in range(m):
    mod.addConstr(z[i] >= sum(b[j] * x.iloc[i, j] for j in range(n - 1)) - y[i])

################################################
# Construct Objective
mod.setObjective((sum(z[i] for i in range(m)) * 1/m ), GRB.MINIMIZE)

################################################
# Update and solve
mod.update()
mod.optimize()

################################################
# Print Optimal Solution
print("Optimal Value:", mod.objVal)
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3400 rows, 1712 columns and 39312 nonzeros
Model fingerprint: 0xefadf137
Coefficient statistics:
  Matrix range     [5e-01, 5e+02]
  Objective range  [6e-04, 6e-04]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 2e+03]
Presolve removed 0 rows and 1 columns

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3400 rows, 1712 columns and 39312 nonzeros
Model fingerprint: 0xefadf137
Coefficient statistics:
  Matrix range     [5e-01, 5e+02]
  Objective range  [6e-04, 6e-04]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 2e+03]
Presolve removed 0 rows and 1 columns
Presolve time: 0.01s
Presolved: 3400 rows, 1711 columns, 39312 nonzeros

Concurrent LP optimizer: primal simplex, dual simplex, and barrier
Showing barrier log only...

Ordering time: 0.00s

Barrier statistics:
 Dense cols : 11
 AA' NZ     : 2.843e+04
 Factor NZ  : 3.107e+04 (roughly 2 MB of memory)
 Factor Ops : 3.757e+05 (less than 1 second per iteration)
 Threads    : 1

                 Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   2.13504409e+03  0.00000000e+00  1.15e+03 0.00e+00  2.72e+01     0s
   1   1.42910837e+03  1.34566856e+00  0.00e+00 2.91e-04  3.02e+00     0s
```

```
   2   1.55937944e+02  4.70911307e+00  0.00e+00 5.51e-05  3.14e-01     0s
   3   4.80279917e+01  1.64581743e+01  0.00e+00 1.39e-05  6.54e-02     0s
   4   3.86594080e+01  3.25188787e+01  0.00e+00 2.78e-17  1.27e-02     0s
   5   3.71696280e+01  3.53267005e+01  0.00e+00 2.78e-17  3.81e-03     0s
   6   3.67521085e+01  3.62795031e+01  0.00e+00 2.78e-17  9.77e-04     0s
   7   3.65976779e+01  3.63932273e+01  0.00e+00 2.78e-17  4.23e-04     0s
   8   3.65358247e+01  3.64472001e+01  0.00e+00 2.78e-17  1.83e-04     0s
   9   3.65025918e+01  3.64835189e+01  0.00e+00 2.78e-17  3.94e-05     0s
  10   3.64936858e+01  3.64921299e+01  0.00e+00 7.22e-13  3.22e-06     0s
  11   3.64925019e+01  3.64923490e+01  0.00e+00 7.79e-13  3.16e-07     0s
  12   3.64923959e+01  3.64923959e+01  0.00e+00 4.29e-13  4.29e-11     0s

Barrier solved model in 12 iterations and 0.05 seconds (0.05 work units)
Optimal objective 3.64923959e+01

Crossover log...

      16 DPushes remaining with DInf 0.0000000e+00                 0s
       0 DPushes remaining with DInf 0.0000000e+00                 0s

       0 PPushes remaining with PInf 0.0000000e+00                 0s

  Push phase complete: Pinf 0.0000000e+00, Dinf 7.7924398e-14      0s


Solved with barrier
Iteration    Objective       Primal Inf.    Dual Inf.      Time
      19    3.6492396e+01   0.000000e+00   0.000000e+00      0s

Use crossover to convert LP symmetric solution to basic solution...
Crossover log...

       0 DPushes remaining with DInf 0.0000000e+00                 0s

       0 PPushes remaining with PInf 0.0000000e+00                 0s

  Push phase complete: Pinf 0.0000000e+00, Dinf 1.0972841e-14      0s

Iteration    Objective       Primal Inf.    Dual Inf.      Time
      33    3.6492396e+01   0.000000e+00   0.000000e+00      0s

Solved in 33 iterations and 0.07 seconds (0.05 work units)
Optimal objective  3.649239588e+01
Optimal Value: 36.49239587815401
```

In [ ]:
```
###############################################
# Print and Calculate the Error
test['price_pred'] = 0
for j in range(n - 1):
    test['price_pred'] += test.iloc[:, j] * b[j].x
    test['price_pred'] = test['price_pred'] + b[n - 1].x

print("Mean Absolute Error:", mean_absolute_error(test['price'], test['price_pred']))
```

```
Mean Absolute Error: 35.64907272288845
```

The prediction error for the test set of our model is ~$35.65 per night.

Model 2. Suppose that to improve interpretability, you wish to build a model that predicts Airbnb prices using only the three most important variables. Modify **Model 1** by including a constraint that allows at most three variables to have non-zero coefficients.

The solution can be given by the following equations:

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} Z_i \tag{5}$$

$$\text{s.t.} \quad Z_i \geq y_i - (\sum_{j=1}^{n} \beta_j \times X_{ij}) \quad i = 1 \dots m \tag{6}$$

$$Z_i \geq (\sum_{j=1}^{n} \beta_j \times X_{ij}) - y_i \quad i = 1 \dots m \tag{7}$$

$$\sum_{j=1}^{n} F_j \leq 3 \tag{8}$$

$$\beta_j \leq F_j \times 1000 \quad j = 1 \dots n \tag{9}$$

$$\beta_j \geq F_j \times (-1000) \quad j = 1 \dots n \tag{10}$$

In [ ]:
```
y = train['price']
m = train.shape[0]
n = train.drop(columns=['price']).shape[1]
x = train.drop(columns=['price'])

###############################################
```

```
# Initialize the model
mod = Model()

################################################
# Define Variables
z = mod.addVars(m)
b = mod.addVars(n)
f = mod.addVars(n, vtype = GRB.BINARY)

################################################
# Define Constraints
for i in range(m):
    mod.addConstr(z[i] >= y[i] - sum(b[j] * x.iloc[i, j] for j in range(n - 1)))

for i in range(m):
    mod.addConstr(z[i] >= sum(b[j] * x.iloc[i, j] for j in range(n - 1)) - y[i])

mod.addConstr(sum(f[j] for j in range(n)) <= 3)

for j in range(n):
    mod.addConstr(b[j] <= f[j] * 1000)

for j in range(n):
    mod.addConstr(b[j] >= f[j] * -1000)

################################################
# Construct Objective
mod.setObjective((sum(z[i] for i in range(m)) * 1/m ), GRB.MINIMIZE)

################################################
# Update and solve
mod.update()
mod.optimize()

################################################
# Print Optimal Solution
print("Optimal Value:", mod.objVal)
for j in range(n):
    print(f'b{j} {b[j].x}')
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3425 rows, 1724 columns and 39372 nonzeros
Model fingerprint: 0xde9bf761
Variable types: 1712 continuous, 12 integer (12 binary)
Coefficient statistics:
  Matrix range     [5e-01, 1e+03]
  Objective range  [6e-04, 6e-04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [3e+00, 2e+03]
Found heuristic solution: objective 144.9682353

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3425 rows, 1724 columns and 39372 nonzeros
Model fingerprint: 0xde9bf761
Variable types: 1712 continuous, 12 integer (12 binary)
Coefficient statistics:
  Matrix range     [5e-01, 1e+03]
  Objective range  [6e-04, 6e-04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [3e+00, 2e+03]
Found heuristic solution: objective 144.9682353
Presolve removed 841 rows and 416 columns
Presolve time: 0.02s
Presolved: 2584 rows, 1308 columns, 29751 nonzeros
Variable types: 1297 continuous, 11 integer (11 binary)

Root relaxation: objective 3.649240e+01, 1416 iterations, 0.11 seconds (0.31 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   36.49240    0    6  144.96824   36.49240  74.8%     -    0s
H    0     0                      86.1956761   36.49240  57.7%     -    0s
H    0     0                      61.6243748   36.49240  40.8%     -    0s
H    0     0                      39.5804163   36.49240  7.80%     -    0s
     0     0   36.49437    0    8   39.58042   36.49437  7.80%     -    0s
     0     0   36.50118    0    9   39.58042   36.50118  7.78%     -    0s
     0     0   36.53006    0    8   39.58042   36.53006  7.71%     -    0s
```

```
     0     0   36.53006    0    9   39.58042   36.53006  7.71%     -    0s
H    0     0                      39.2200000   36.53006  6.86%     -    0s
     0     0   37.29731    0    9   39.22000   37.29731  4.90%     -    0s
H    0     0                      38.3388235   37.29731  2.72%     -    0s
     0     0   37.32781    0    9   38.33882   37.32781  2.64%     -    0s
     0     0   38.10807    0    9   38.33882   38.10807  0.60%     -    0s
     0     0   38.15628    0    9   38.33882   38.15628  0.48%     -    0s

Cutting planes:
  Gomory: 1
  MIR: 1
  Flow cover: 37

Explored 1 nodes (1584 simplex iterations) in 0.49 seconds (0.84 work units)
Thread count was 8 (of 8 available processors)

Solution count 6: 38.3388 39.22 39.5804 ... 144.968

Optimal solution found (tolerance 1.00e-04)
Best objective 3.833882352941e+01, best bound 3.833882352941e+01, gap 0.0000%
Optimal Value: 38.33882352941181
b0 0.0
b1 0.0
b2 52.0
b3 14.0
b4 0.0
b5 32.0
b6 0.0
b7 0.0
b8 0.0
b9 0.0
b10 0.0
b11 0.0
```

a) List the names and coefficients of the three variables selected by the optimization model.

In [ ]:
```python
# List the names and coefficients of the three variables selected by the optimization model.
print("The three variables selected by the optimization model are:")
for j in range(n):
    if f[j].x == 1:
        print(f'Variable {j} is "{train.columns[j]}" with coefficient {b[j].x}')
```

```
The three variables selected by the optimization model are:
Variable 2 is "Entire home" with coefficient 52.0
Variable 3 is "accommodates" with coefficient 14.0
Variable 5 is "bedrooms" with coefficient 32.0
```

b) What is the new prediction error, in $/night, of **Model 2**?

In [ ]:
```python
# Print and Calculate the Error
test['price_pred'] = 0
for j in range(n - 1):
    test['price_pred'] += test.iloc[:, j] * b[j].x
    test['price_pred'] = test['price_pred'] + b[n - 1].x

print("Mean Absolute Error:", mean_absolute_error(test['price'], test['price_pred']))
```

```
Mean Absolute Error: 37.73676680972818
```

The prediction error for the test set of our model is ~$37.73 per night.

Model 3. Suppose now you wish to build a model that predicts Airbnb listing price using only three variables, where one of the variables is the number of beds.

The solution can be given by the following equations:

$$\text{minimize} \quad \frac{1}{m}\sum_{i=1}^{m} Z_i \tag{11}$$

$$\text{s.t.} \quad Z_i \geq y_i - (\sum_{j=1}^{n} \beta_j \times X_{ij}) \quad i = 1 \dots m \tag{12}$$

$$Z_i \geq (\sum_{j=1}^{n} \beta_j \times X_{ij}) - y_i \quad i = 1 \dots m \tag{13}$$

$$\sum_{j=1}^{n} F_j \leq 3 \tag{14}$$

$$\beta_j \leq F_j \times 1000 \quad j = 1 \dots n \tag{15}$$

$$\beta_j \geq F_j \times (-1000) \quad j = 1 \dots n \tag{16}$$

$$F_6 = 1 \tag{17}$$

In [ ]:

```
y = train['price']
m = train.shape[0]
n = train.drop(columns=['price']).shape[1]
x = train.drop(columns=['price'])

################################################
# Initialize the model
mod = Model()

################################################
# Define Variables
z = mod.addVars(m)
b = mod.addVars(n)
f = mod.addVars(n, vtype = GRB.BINARY)

################################################
# Define Constraints
for i in range(m):
    mod.addConstr(z[i] >= y[i] - sum(b[j] * x.iloc[i, j] for j in range(n - 1)))

for i in range(m):
    mod.addConstr(z[i] >= sum(b[j] * x.iloc[i, j] for j in range(n - 1)) - y[i])

mod.addConstr(sum(f[j] for j in range(n)) <= 3)

for j in range(n):
    mod.addConstr(b[j] <= f[j] * 1000)

for j in range(n):
    mod.addConstr(b[j] >= f[j] * -1000)

mod.addConstr(f[6] == 1)

################################################
# Construct Objective
mod.setObjective((sum(z[i] for i in range(m)) * 1/m ), GRB.MINIMIZE)

################################################
# Update and solve
mod.update()
mod.optimize()

################################################
```

```python
# Print Optimal Solution
print("Optimal Value:", mod.objVal)
for j in range(n):
    print(f'b{j} {b[j].x}')
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3426 rows, 1724 columns and 39373 nonzeros
Model fingerprint: 0x7002e2af
Variable types: 1712 continuous, 12 integer (12 binary)
Coefficient statistics:
  Matrix range     [5e-01, 1e+03]
  Objective range  [6e-04, 6e-04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
Found heuristic solution: objective 144.9682353

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 3426 rows, 1724 columns and 39373 nonzeros
Model fingerprint: 0x7002e2af
Variable types: 1712 continuous, 12 integer (12 binary)
Coefficient statistics:
  Matrix range     [5e-01, 1e+03]
  Objective range  [6e-04, 6e-04]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
Found heuristic solution: objective 144.9682353
Presolve removed 843 rows and 417 columns
Presolve time: 0.02s
Presolved: 2583 rows, 1307 columns, 29748 nonzeros
Variable types: 1297 continuous, 10 integer (10 binary)

Root relaxation: objective 3.649240e+01, 1416 iterations, 0.11 seconds (0.31 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   36.49240    0    7  144.96824   36.49240  74.8%     -    0s
H    0     0                     105.3365564   36.49240  65.4%     -    0s
H    0     0                      48.6558384   36.49240  25.0%     -    0s
     0     0   36.49640    0    9   48.65584   36.49640  25.0%     -    0s
     0     0   36.50917    0    9   48.65584   36.50917  25.0%     -    0s
H    0     0                      44.5017647   36.50917  18.0%     -    0s
     0     0   36.88719    0    8   44.50176   36.88719  17.1%     -    0s
```

```
H    0     0                    40.0730147   36.88719  7.95%     -    0s
     0     0   36.88719    0    8   40.07301   36.88719  7.95%     -    0s
     0     0   37.32781    0    9   40.07301   37.32781  6.85%     -    0s
     0     0   37.32781    0    9   40.07301   37.32781  6.85%     -    0s
     0     0   38.84881    0    8   40.07301   38.84881  3.05%     -    0s
     0     0   39.13919    0    8   40.07301   39.13919  2.33%     -    0s

Cutting planes:
  Gomory: 2
  MIR: 8
  Flow cover: 53

Explored 1 nodes (1638 simplex iterations) in 0.54 seconds (0.91 work units)
Thread count was 8 (of 8 available processors)

Solution count 5: 40.073 44.5018 48.6558 ... 144.968

Optimal solution found (tolerance 1.00e-04)
Best objective 4.007301470588e+01, best bound 4.007301470588e+01, gap 0.0000%
Optimal Value: 40.073014705882386
b0 0.0
b1 0.0
b2 67.875
b3 0.0
b4 0.0
b5 47.375
b6 12.125
b7 0.0
b8 0.0
b9 0.0
b10 0.0
b11 0.0
```

a) List the names and coefficients of the two other variables selected by the optimization model.

In [ ]:
```python
# List the names and coefficients of the three variables selected by the optimization model.
print("The three variables selected by the optimization model are:")
for j in range(n):
    if f[j].x == 1:
        print(f'Variable {j} is "{train.columns[j]}" with coefficient {b[j].x}')
```

```
The three variables selected by the optimization model are:
Variable 2 is "Entire home" with coefficient 67.875
Variable 5 is "bedrooms" with coefficient 47.375
Variable 6 is "beds" with coefficient 12.125
```

b) Which variable was in **Model 2** but is no longer in **Model 3**? Briefly explain in 1-2 sentences why this variable might have been dropped.

Variable 3 which is "accomodates" was dropped from the model because it has the lowest predictive power out of the three choices between "entire home", "accomodates" and "bedrooms". Therefore "bedrooms" and "entire home" is the best combination of two variables that along with "beds" minimise the mean absolute error.

c) What is the new prediction error, in $/night, of **Model 3**?

In [ ]:
```
# Print and Calculate the Error
test['price_pred'] = 0
for j in range(n - 1):
    test['price_pred'] += test.iloc[:, j] * b[j].x
    test['price_pred'] = test['price_pred'] + b[n - 1].x

print("Mean Absolute Error:", mean_absolute_error(test['price'], test['price_pred']))
```

```
Mean Absolute Error: 38.59960658082976
```

The prediction error for the test set of our model is ~$38.60 per night.