Game Theory Models:

Wald (Pessimistic), Laplace, Hurwicz, Benefit, Wald (Optimistic)

In [1]:
```
from gurobipy import *
import numpy as np
import pandas as pd
```

In [2]:
```
# Read dataset
df = pd.read_csv('df_final_first_3.csv')
data = pd.read_csv('/Users/hpone/Desktop/NUS MSBA/DBA5103/Term project/Mansi code/dba5103_gp-widya/df_model_matrix
data.head()
```

Out[2]:

	datetime	index	year_id	timeperiod_id	commodity_desc	amount	price	benefit_criterion	regret_criterion	lower_bound	upper_bound
0	2004-01-31	216	2004	1	Catfish	53849.0	66.8	0.0	6.0	52.838403	85.460174
1	2004-02-29	217	2004	2	Catfish	54173.0	70.3	3.5	2.5	59.907307	88.568631
2	2004-03-31	218	2004	3	Catfish	60272.0	72.3	5.5	0.5	66.034410	89.013121
3	2004-04-30	219	2004	4	Catfish	53896.0	72.8	6.0	0.0	70.573322	87.091338
4	2004-05-31	220	2004	5	Catfish	52324.0	72.0	5.2	0.8	70.665213	80.345013

In [3]:

```
# initial all criterion matrices

# game or neutral matrix
wald_matrix = df['price'].values.reshape(9,12)
# pessistic/optimistic matrix ~ lower/upper bounds respectively
pessimistic_matrix = data['lower_bound'].values.reshape(9,12)
optimistc_matrix = data['upper_bound'].values.reshape(9,12)
benefit_matrix = df['benefit_criterion'].values.reshape(9,12)
regret_matrix = df['regret_criterion'].values.reshape(9,12)

alpha = 0.80
hurwicz_matrix = optimistc_matrix*alpha + (1-alpha)*pessimistic_matrix
laplace_matrix = 0.5*optimistc_matrix + 0.5*pessimistic_matrix

# Number of years: M; No. of months: N = 12
M, N = wald_matrix.shape

month_dict = {0:"Jan", 1:"Feb", 2:"Mar", 3:"Apr", 4:"May", 5:"Jun", 6:"Jul",7:"Aug",8:"Sept",9:"Oct",10:"Nov",11:"

# monthly mean for all years combined
monthly_mean = [np.mean(wald_matrix[:,i]) for i in range(N)]
monthly_mean
```

Out[3]:

```
[81.86666666666666,
 82.84444444444445,
 84.62222222222222,
 85.67777777777778,
 84.65555555555554,
 83.8,
 82.78888888888888,
 82.33333333333333,
 82.43333333333334,
 82.52222222222223,
 82.4888888888889,
 82.55555555555556]
```

In [4]:
```
# Setup Criterion Based Linear Programming Optimization Model
def model_setup(name, matrix):
    # initialize criterion model
    model = Model(f"{name} Criterion")

    # Decision Variables for percentage of catfish sells every month
    p = model.addVars(N)
    # Decision Variable for Price per unit of catfish ~ cents/pounds
    Z = model.addVar(name = 'Z')

    # Set objective to maximize Price per unit
    model.setObjective(Z, GRB.MAXIMIZE)

    for i in range(M):
        # Contraints for Sells every year to be greater than the optimized result
        model.addConstr(quicksum(matrix[i, j]*p[j] for j in range(N)) >= Z, 'Contraints')
        # percentages for every year add up to 1
        model.addConstr (quicksum(p[j] for j in range(N)) == 1)

    model.optimize()

    return model
```

Pessimistic Wald Criterion Model Optimization

In [5]:
```
# Pessimistic Matrix with Wald Criterion Model Optimization
model_name = "Pessimistic Wald"
pessimistic_wald_model =  model_setup(model_name, pessimistic_matrix)

#  Print optimal sells for every month
print("\n Optimal solution:")
price = 0
for i, v in enumerate(pessimistic_wald_model.getVars()[:N]):
    print(v.VarName, v.x)

# Optimal Price given by model
pessimistic_price = round(pessimistic_wald_model.objVal, 3)
print('{} Criterion Z objective => Price: {} cents/pound'.format(model_name, round(pessimistic_wald_model.objVal,
```

```
Academic license - for non-commercial use only - expires 2021-12-22
Using license file /Users/hpone/gurobi.lic
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (mac64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 18 rows, 13 columns and 225 nonzeros
Model fingerprint: 0x3656bf2e
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 8 rows and 0 columns
Presolve time: 0.01s
Presolved: 10 rows, 13 columns, 129 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    8.0698122e+02   8.350033e+02   0.000000e+00      0s
       3    7.0665213e+01   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.03 seconds
Optimal objective  7.066521265e+01

 Optimal solution:
C0 0.0
C1 0.0
C2 0.0
C3 0.0
C4 1.0
C5 0.0
C6 0.0
C7 0.0
C8 0.0
C9 0.0
C10 0.0
C11 0.0
Pessimistic Wald Criterion Z objective => Price: 70.665 cents/pound
```

Pessimistic Wald Criterion Optimal Solution:

p4 = 1 and Maximum Z = 70.665

The solution indicates that out of the total catfish supplied to middlemen, 100% should be sold in the month of May. Thus the guaranteed average price received by the catfish producers(farmers) will be 70.665 cents/pound

Laplace Criterion

In [6]:

```
# Laplace Criterion Model Optimization
model_name = "Laplace"
laplace_model =  model_setup(model_name, laplace_matrix)

#  Print optimal sells for every month
print("\n Optimal solution:")
for i, v in enumerate(laplace_model.getVars()[:N]):
    print(v.VarName, v.x)

# Optimal Price given by model
laplace_price = round(laplace_model.objVal, 3)
print('{} Criterion Z objective => Price : {} cents/pound'.format(model_name, round(laplace_model.objVal, 3)))
```

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (mac64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 18 rows, 13 columns and 225 nonzeros
Model fingerprint: 0xb96521ad
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 8 rows and 0 columns
Presolve time: 0.02s
Presolved: 10 rows, 13 columns, 129 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.2858272e+02   8.993544e+02   0.000000e+00      0s
       4    7.8528060e+01   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.03 seconds
Optimal objective  7.852806012e+01

 Optimal solution:
C0 0.0
C1 0.0
C2 0.23252181968706165
C3 0.7674781803129384
C4 0.0
C5 0.0
C6 0.0
C7 0.0
C8 0.0
C9 0.0
C10 0.0
C11 0.0
Laplace Criterion Z objective => Price : 78.528 cents/pound
```

Laplace Criterion Optimal Solution:

**p2 = 0.023 and p3=0.77

0.023 *Monthly_mean_for_March + 0.767* Monthly_mean_for_April = 78.528 cents/pound

The optimal solution indicates that out of the total catfish supplied to middlemen, 2.3 and 76.7 percent of catfish should be sold in the months of March and April respectively. Thus the guaranteed average price received by the catfish producers(farmers) will be 78.528 cents/pound

Hurwicz

In [7]:

```
# Hurwicz Criterion Model Optimization
model_name = "Hurwicz"
hurwicz_model =  model_setup(model_name, hurwicz_matrix)

#  Print optimal sells for every month
print("\n Optimal solution:")
for i, v in enumerate(hurwicz_model.getVars()[:N]):
    print(v.VarName, v.x)

# Optimal Price given by model
hurwicz_price = round(hurwicz_model.objVal, 3)
print('{} Criterion Z objective => Price : {} cents/pound'.format(model_name, round(hurwicz_model.objVal, 3)))
```

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (mac64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 18 rows, 13 columns and 225 nonzeros
Model fingerprint: 0x9da72ce5
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 8 rows and 0 columns
Presolve time: 0.04s
Presolved: 10 rows, 13 columns, 129 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.0015436e+03   9.088044e+02   0.000000e+00      0s
       6    8.4417379e+01   0.000000e+00   0.000000e+00      0s

Solved in 6 iterations and 0.05 seconds
Optimal objective  8.441737908e+01

 Optimal solution:
C0 0.0
C1 0.0
C2 1.0
C3 0.0
C4 0.0
C5 0.0
C6 0.0
C7 0.0
C8 0.0
C9 0.0
C10 0.0
C11 0.0
Hurwicz Criterion Z objective => Price : 84.417 cents/pound
```

Hurwicz Criterion Optimal Solution:

p2 = 1.0

1 * Monthly_mean_for_March = 84.417 cents/pound

The optimal solution indicates that out of the total catfish supplied to middlemen, 100 percent of catfish should be sold in the month of March. Thus the guaranteed average price received by the catfish producers(farmers) will be 84.417 cents/pound

Benefit Criterion

In [8]:

```
# Benefit Criterion Model Optimization
model_name = 'Benefit'
benefit_model =  model_setup("Benefit", benefit_matrix)

#  Print optimal sells for every month
benefit_price = 0
for i, v in enumerate(benefit_model.getVars()[:N]):
    if v.x > 0:
        benefit_price = benefit_price + monthly_mean[i]*v.x
    print(v.VarName, v.x)

# Optimal Price given by model
print('{} Criterion Z objective : {}'.format(model_name, round(benefit_model.objVal, 3)))
print("Price given by Benefit Criterion : : {} cents/pound".format(round(benefit_price,3)))
```

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (mac64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 18 rows, 13 columns and 216 nonzeros
Model fingerprint: 0x36faad9b
Coefficient statistics:
  Matrix range     [1e-01, 5e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 8 rows and 0 columns
Presolve time: 0.02s
Presolved: 10 rows, 13 columns, 120 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.0511500e+01   1.446725e+01   0.000000e+00      0s
       4    1.1822222e+00   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.02 seconds
Optimal objective  1.182222222e+00
C0 0.022222222222224117
C1 0.0
C2 0.9777777777777759
C3 0.0
C4 0.0
C5 0.0
C6 0.0
C7 0.0
C8 0.0
C9 0.0
C10 0.0
C11 0.0
Benefit Criterion Z objective : 1.182
Price given by Benefit Criterion : : 84.561 cents/pound
```

Benefit Wald Criterion Optimal Solution:

p0 = 0.022 and p2=0.978 Maximum Z = 1.182

0.022 *Monthly_mean_for_January + 0.978* Monthly_mean_for_March = 84.561 cents/pound

The optimal solution indicates that out of the total catfish supplied to middlemen, 2.2 and 97.8 percent of catfish should be sold in the months of January and March respectively. Thus the guaranteed average price received by the catfish producers(farmers) will be 84.561 cents/pound

Optimistic Wald

In [9]:

```
# Optimistic Matrix with Wald Criterion Model Optimization
model_name = "Optimistic Wald"
optimistc_wald_model =  model_setup(model_name, optimistc_matrix)

#  Print optimal sells for every month
print("\n Optimal solution:")
for i, v in enumerate(optimistc_wald_model.getVars()[:N]):
    print(v.VarName, v.x)

# Optimal Price given by model
optimistic_price = round(optimistc_wald_model.objVal, 3)
print('{} Criterion Z objective => Price : {} cents/pound'.format(model_name, round(optimistc_wald_model.objVal, 3
```

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (mac64)
Thread count: 2 physical cores, 4 logical processors, using up to 4 threads
Optimize a model with 18 rows, 13 columns and 225 nonzeros
Model fingerprint: 0x1d346ea2
Coefficient statistics:
  Matrix range     [1e+00, 2e+02]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve removed 8 rows and 0 columns
Presolve time: 0.01s
Presolved: 10 rows, 13 columns, 129 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.0501842e+03   9.497797e+02   0.000000e+00      0s
       3    8.9013121e+01   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.02 seconds
Optimal objective  8.901312148e+01

 Optimal solution:
C0 0.0
C1 0.0
C2 1.0
C3 0.0
C4 0.0
C5 0.0
C6 0.0
C7 0.0
C8 0.0
C9 0.0
C10 0.0
C11 0.0
Optimistic Wald Criterion Z objective => Price : 89.013 cents/pound
```

Optimistic Wald Criterion Optimal Solution:

p2 = 1 and Maximum Z = 89.013

The solution indicates that out of the total catfish supplied to middlemen, 100% should be sold in the month of March. Thus the guaranteed averaGe price received by the catfish producers(farmers) will be 89.013 cents/pound

Consolidating Results

In [10]:
```
results_dict = {
    "pessimistic": pessimistic_price,
    "laplace":laplace_price,
    "hurwicz":hurwicz_price,
    "benefit":benefit_price,
    "optimistic":optimistic_price,
}
```

In [11]:
```
results_dict
```

Out[11]:
```
{'pessimistic': 70.665,
 'laplace': 78.528,
 'hurwicz': 84.417,
 'benefit': 84.56098765432097,
 'optimistic': 89.013}
```

In [12]:
```
difference = {}
for key, value in results_dict.items():
    difference[key] = (1 + (results_dict[key] - pessimistic_price) / results_dict[key])*100-100
difference
```

Out[12]:
```
{'pessimistic': 0.0,
 'laplace': 10.012988997554999,
 'hurwicz': 16.290557589111202,
 'benefit': 16.433095260342427,
 'optimistic': 20.612719490411507}
```

In [13]:
```
results_df = pd.DataFrame(index=["Price (\xa2 per lb)", 'Improvement %'], data=[results_dict, difference])
results_df
```

Out[13]:

	pessimistic	laplace	hurwicz	benefit	optimistic
Price (¢ per lb)	70.665	78.528000	84.417000	84.560988	89.013000
Improvement %	0.000	10.012989	16.290558	16.433095	20.612719