Sprawozdanie z projektu 25102 z WDWR Dawid Bartosiak

1. Zaproponować jednokryterialny model wyboru w warunkach ryzyka z wartością średnią jako miarą zysku. Wyznaczyć rozwiązanie optymalne.

1.1. Początkowe obserwacje

Zauważyć można że nasze kryterium jakości opiera się o wartość średnią. Z tego powodu w tym zadaniu nie jest konieczne generowanie scenariuszy, lecz możliwe jest użycie wartości oczekiwanej. Pozwala to w tym przypadku na znaczne zredukowanie złożoności modelu dla tego zadania. Mając informację o rozkładzie mogę przejść do obliczenia wartości oczekiwanych, korzystam tutaj ze wzoru na wartość oczekiwaną zawężonego rozkładu t-Studenta:

$$\mathrm{E}(R) = \mu + \sigma \frac{\Gamma(\frac{\mathrm{v}-1}{2})\left((v+a^2)^{-\frac{v-1}{2}} - (v-b^2)^{-\frac{v-1}{2}}\right)\mathrm{v}^{\frac{\mathrm{v}}{2}}}{2\left(F_v(b) - F_v(a)\right)\Gamma\left(\frac{v}{2}\right)\Gamma\left(\frac{1}{2}\right)},$$

gdzie w naszym przypadku:

$$v = 5, a = 80, b = 120,$$

$$\mu = \begin{bmatrix}116\\102\\113\\100\\107\\110\end{bmatrix}, \qquad \sigma = \begin{bmatrix}1\\6\\2\\7\\4\\3\end{bmatrix},$$

zatem stosując taką samą kolejność jak w wektorach μ i σ:

$$\mathrm{E}(R) \approx \begin{bmatrix}116\\102\\113\\100\\107\\110\end{bmatrix},$$

Opisane obliczenia mogły także zostać pominięte z uwagi na to jak wyglądają parametry μ i σ, ponieważ zakresy rozkładu zostały przycięte w odległościach $\geq 3 \cdot \sigma$, poza jednym z parametrów, co pozwala nam z reguły odciąć te wartości powołując się na regułę 3 sigm, jednak dla pewności obliczenia zostały wykonane w MATLAB, co potwierdziło, że średnie są dalej takie same.

kod w MATLAB który wykonał obliczenia:

```
v = 5;
a = 80;
b = 120;

mu = [116; 102; 113; 100; 107; 110];
sigma = [1; 6; 2; 7; 4; 3];

gamma_v2 = gamma(v/2);
gamma_v12 = gamma((v-1)/2);
gamma_12 = gamma(1/2);

F_v_a = tcdf((a - mu) ./ sigma, v);
F_v_b = tcdf((b - mu) ./ sigma, v);

term1 = (v + a.^2).^(-(v-1)/2);
term2 = (v + b.^2).^(-(v-1)/2);
numerator = gamma_v12 * (term1 - term2) * v^(v/2);
denominator = 2 * (F_v_b - F_v_a) * gamma_v2 * gamma_12;

E_R = mu + sigma .* (numerator ./ denominator);

disp('Wartość oczekiwana E(R):');
disp(E_R);
```

Rys. 1. Kod matlab do obliczenia wartości oczekiwanych.

1.2. Model matematyczny problemu

Celem zoptymalizowania postawionego zadania konieczne jest opisanie problemu w sposób matematyczny. Przy oznaczeniach:

Zbiory:

typy – lista typów olejów

miesiące – lista miesięcy

Parametry:

- s[typ oleju, miesiąc] – stan magazynowy oleju danego typu w danym miesiącu ≥ 0
- hardness[typ oleju] – twardość oleju danego typu ≥ 0
- price[typ oleju, miesiąc] – cena oleju danego typu w danym miesiącu ≥ 0

Zmienne:

- x[typ oleju, miesiąc] – ilość rafinowanego oleju danego typu w danym miesiącu ≥ 0
- b[typ oleju, miesiąc] – ilość zakupionego oleju danego typu w danym miesiącu ≥ 0
- tot[miesiąc] – pomocnicza – całkowita produkcja w danym miesiącu ≥ 0
- y[typ oleju, miesiąc] – pomocnicza – czy olej danego typu został użyty w produkcji w danym miesiącu – binarna

Ograniczenia które musi spełnić model prezentują się następująco:

- Ograniczenie na twardość produktu gotowego:

$$\bigwedge_{m}^{miesiące} 3 \cdot tot[m] \leq \sum_{t}^{typ\ oleju} hardness[t] \leq 6 \cdot tot[m]$$

- Ograniczenie na rafinację roślinnego oleju w danym miesiącu:

$$\bigwedge_{m}^{miesiące} x[\mathrm{A}, m] + x[\mathrm{B}, m] \leq 220$$

- Ograniczenie na rafinację nieroślinnego oleju w danym miesiącu:

$$\bigwedge_{m}^{miesiące} x[\mathrm{C}, m] \leq 270$$

- Ograniczenie na wykorzystanie maksymalnie 2 rodzajów oleju w danym miesiącu:

$$\bigwedge_{m}^{miesiące} \sum_{t}^{typy} y[t, m] \leq 2$$

Ograniczenie na maksymalny poziom magazynowy zostało uwzględnione na podstawie ograniczenia dziedziny zmiennej s. Każda zmienna użyta w zadaniu dodatkowo jest większa od 0.

Powiązanie parametrów x, tot i y:

$$\bigwedge_{t}^{typy} \bigwedge_{m}^{miesiące} x[t, m] \leq M \cdot y[t, m] \qquad \bigwedge_{m}^{miesiące} tot[m] = \sum_{t}^{typy} x[t, m]$$

Obliczenie stanu magazynowego s:

$$\bigwedge_{t}^{typy} \bigwedge_{m}^{miesiące} s[t, m] = \begin{cases} stan\ początkowy, & pierwszy\ miesiąc \\ s[t, prev(m)] + b[t, m] - x[t, m], & kolejny\ miesiąc \end{cases}$$

Otrzymana funkcja celu wygląda następująco:

$$profit = 170 \cdot \sum_{m}^{miesiące} tot[m] - \sum_{t}^{typy} \sum_{m}^{miesiące} price[t, m] \cdot b[t, m] - 10 \cdot s[t, m]$$

i jest ona maksymalizowana.

Otrzymane przeze mnie optymalne działanie to:

- Styczeń – dokupić 20 jednostek oleju B i 70 jednostek oleju C, następnie dokonać rafinacji 220 jednostek oleju B i 270 jednostek oleju C.
- Luty – dokupić 20 jednostek oleju A i 270 jednostek oleju C, następnie dokonać rafinacji 220 jednostek oleju A i 270 jednostek oleju C.

Końcowy stan magazynowy to w pełni puste magazyny każdego z olejów oraz przychód w wysokości 122950zł. Kod programu ampl wykonującego optymalizację przedstawiam na następnych stronach (Rys. 2 oraz Rys. 3):

```
# Zbiory
set TYPES;
set MONTHS ordered;

# Parametry
param sell_price>=0;        # cena sprzedaży produktu [zł/tonę]
param stor_cost>=0;         # koszt magazynowania [zł/tonę]
param stor_cap>=0;          # maksymalny stan magazynowy [ton]
param init_inv {TYPES}>=0;  # początkowy stan magazynowy dla każdego rodzaju [ton]

# Ceny surowego oleju (średnie) - miesiąc Jan i Feb
param price {TYPES, MONTHS}>=0;

# Limity rafinacji:
param cap_plant>=0;         # oleje roślinne: A, B
param cap_nonplant>=0;      # olej nieroślinny: C
param M>=0;                 # duża stała
param hardness {TYPES}>=0;  # Współczynnik twardości

# Zmienna pomocnicza - całkowita produkcja w miesiącu m
var tot {MONTHS} >= 0;
# Zmienne decyzyjne:
var y {TYPES, MONTHS} binary;               # użycie oleju w rafinacji
var b {TYPES, MONTHS} >= 0;                 # zakup surowego oleju
var x {TYPES, MONTHS} >= 0;                 # ilość rafinowanego oleju
var s {TYPES, MONTHS} >= 0, <= stor_cap;    # stan magazynowy


subject to TotalProduction {m in MONTHS}:
    tot[m] = sum {t in TYPES} x[t, m];
# Ograniczenie twardości produktu:
subject to Hardness {m in MONTHS}:
    (3 * tot[m] <= sum {t in TYPES} hardness[t] * x[t, m]) &&
    (sum {t in TYPES} hardness[t] * x[t, m] <= 6 * tot[m]);

# Bilans magazynowy:
subject to InvBalance {t in TYPES, m in MONTHS}:
    s[t, m] = (if m = first(MONTHS) then init_inv[t] else s[t, prev(m)]) + b[t, m] - x[t, m];

# Ograniczenia rafinacji:
subject to RefinePlant {m in MONTHS}:
    x["A", m] + x["B", m] <= cap_plant;

subject to RefineNonPlant {m in MONTHS}:
    x["C", m] <= cap_nonplant;

# Ograniczenie na 2 rodzaje oleju:
subject to AtMostTwoTypes {m in MONTHS}:
    sum {t in TYPES} y[t, m] <= 2;

# Powiązanie zmiennych x i y - jeśli olej t nie jest użyty, to x[t,m] musi być 0:
subject to LinkXY {t in TYPES, m in MONTHS}:
    x[t, m] <= M * y[t, m];

# Funkcja celu
maximize Profit:
    sell_price * sum {m in MONTHS}(tot[m])
    - sum {t in TYPES, m in MONTHS}(price[t, m] * b[t, m] + stor_cost * s[t, m]);
```

Rys. 2. Kod AMPL do zadania 1 – logika.

```
# Zbiory
set TYPES := A B C;
set MONTHS := Jan Feb;

# Parametry
param sell_price := 170;    # cena sprzedaży produktu [zł/tonę]
param stor_cost := 10;      # koszt magazynowania [zł/tonę]
param stor_cap := 800;      # maksymalny stan magazynowy [ton]
param: init_inv :=          # początkowy stan magazynowy dla każdego rodzaju [ton]
                A 200
                B 200
                C 200;

# Ceny surowego oleju (średnie) - styczen i luty
param price: Jan Feb :=
          A 116 100
          B 102 107
          C 113 110;

# Limity rafinacji:
param cap_plant := 220;     # oleje roślinne: A, B
param cap_nonplant := 270;  # olej nieroślinny: C

# Współczynniki twardości:
param: hardness :=
   A 8.4
   B 6.2
   C 2.0;
param M := 10000;  # duża stała
```

Rys. 3. Kod AMPL do zadania 1 – dane.

2. Zaproponować dwukryterialny model zysku i ryzyka z wartością średnią jako miarą zysku i średnią różnicą Giniego jako miarą ryzyka. Dla decyzji $x \in Q$ średnia różnica Giniego jest definiowana jako $\Gamma(x) = \frac{1}{2}\sum_{t'=1}^{T}\sum_{t''=1}^{T}|r_{t'}(x)r_{t''}(x)|\, p_{t'}p_{t''}$, gdzie $r_t(x)$ oznacza realizację dla scenariusza t, p_t prawdopodobieństwo scenariusza t.

Celem realizacji zadań postawionych w tej części projektu rozszerzyłem model z pierwszego zadania poprzez zmianę funkcji celu:

$$profit = 170 \cdot \sum_{m}^{miesiące} tot[m] - \sum_{t}^{typ\ oleju} \sum_{m}^{miesiące} price[t,m] \cdot b[t,m] - 10 \cdot s[t,m]$$

poprzez rozbicie jej na oczekiwany przychód:

$$ExpProfit = \sum_{scenariusz}^{scenariusze} profit[scenariusz] \cdot p[scenariusz]$$

następnie obliczenie ryzyka zgodnie ze wzorem z zadania, z użyciem zmiennej pomocniczej:

p[sc] – prawdopodobieństwo scenariusza sc ≥ 0

d[sc1, sc2] – zmienna pomocnicza wskazująca nieujemną część różnicy między sc1 i sc2 ≥ 0

$$\bigwedge_{sc1}^{scenariusze} \bigwedge_{sc2}^{scenariusze: sc1 != sc2} d[sc1, sc2] \geq profit[sc1] - profit[sc2]$$

$$risk = \frac{1}{2} \sum_{sc1}^{scenariusze} \sum_{sc2}^{scenariusze} p[sc1] \cdot p[sc2] \cdot d[sc1, sc2]$$

oraz przedstawienie funkcji celu:

$$MultObjective = w_{profit} \cdot ExpProfit + w_{risk} \cdot risk$$

Generowanie scenariuszy odbywa się z użyciem skryptu w Pythonie (stosuję 400 scenariuszy z uwagi że jest to największa ilość z czasem obliczeń około 5 minut):

```python
import numpy as np
import scipy.stats as stats

def generate_scenarios(n_scenarios=10, df=5):
    mu = np.array([116, 102, 113, 100, 107, 110])
    Sigma = np.array([
        [1, 1, 0, 1, 1, 1],
        [1, 36, -1, -1, -3, -5],
        [0, -1, 4, 2, 2, 0],
        [1, -1, 2, 49, -5, -2],
        [1, -3, 2, -5, 16, -2],
        [1, -5, 0, -2, -2, 9]
    ])

    t_samples = np.random.standard_t(df, size=(n_scenarios, 6))

    L = np.linalg.cholesky(Sigma)
    correlated_samples = mu + (t_samples @ L.T) / np.sqrt(df / (df - 2))

    constrained_samples = np.clip(correlated_samples, 80, 120)

    ampl_format = "param price :=\n"
    for i in range(n_scenarios):
        scenario = f"S{i+1}"
        for j, (oil, month) in enumerate([("A", "Jan"), ("A", "Feb"), ("B", "Jan"), ("B", "Feb"), ("C", "Jan"), ("C", "Feb")]):
            ampl_format += f"    {oil} {month} {scenario} {int(constrained_samples[i, j])}\n"
    ampl_format += ";\n\n"

    ampl_format += "param p :=\n"
    for i in range(n_scenarios):
        ampl_format += f"    S{i+1} {1/n_scenarios}\n"
    ampl_format += ";"
    ampl_format += "\nset SCENARIOS :="
    for i in range(n_scenarios):
        ampl_format += f" S{i+1}"
    ampl_format += ";"
    return ampl_format

scenarios_ampl = generate_scenarios(100)

with open("scenarios.dat", "w") as file:
    file.write(scenarios_ampl)
```

Rys. 4. Kod w Pythonie do generowania scenariuszy

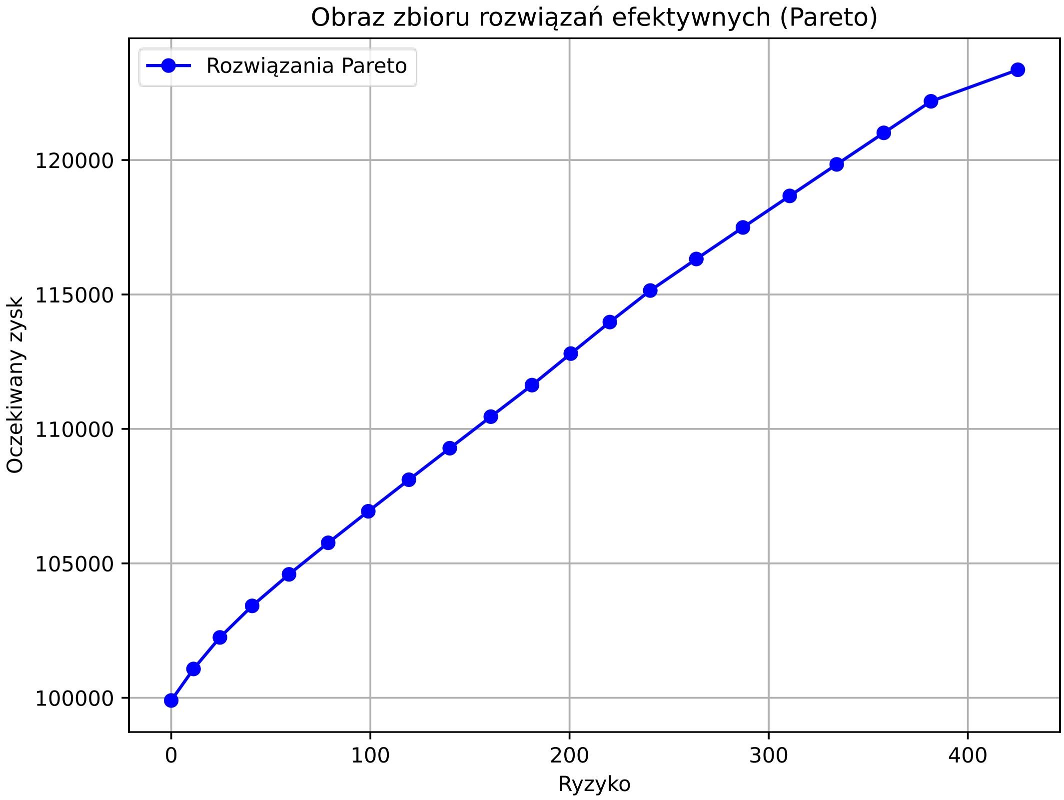
```
# Definicja zysku dla każdego scenariusza s:
var profit {SCENARIOS};
subject to ProfitDefinition {sc in SCENARIOS}:
    profit[sc] = sell_price * sum {m in MONTHS} tot[m]
                 - sum {t in TYPES, m in MONTHS} (price[t, m, sc] * b[t, m] + stor_cost * s[t, m]);

# Obliczenie oczekiwanego zysku:
var ExpProfit = sum {sc in SCENARIOS} p[sc] * profit[sc];

# Zmienne pomocnicze do obliczenia różnicy Giniego bez użycia abs()
var d {SCENARIOS, SCENARIOS} >= 0;

# Ograniczenia dla zmiennych pomocniczych d
subject to DConstraint {sc1 in SCENARIOS, sc2 in SCENARIOS: sc1 != sc2}:
    d[sc1, sc2] >= profit[sc1] - profit[sc2];

# Definicja miary ryzyka (średnia różnica Giniego) bez użycia abs():
var Risk;
subject to GiniDefinition:
    Risk = (1/2) * sum {sc1 in SCENARIOS, sc2 in SCENARIOS} p[sc1] * p[sc2] * d[sc1, sc2];

# Metoda ważona – łączymy oba kryteria z wagami
param w_profit >= 0;
param w_risk >= 0;

# Funkcja celu – chcemy maksymalizować zysk i minimalizować ryzyko
maximize MultiObjective:
    w_profit * ExpProfit - w_risk * Risk;
```

Rys. 5. Zmieniona część kodu w AMPL odpowiedzialna za działanie w nowych warunkach

a. Wyznaczyć obraz zbioru rozwiązań efektywnych w przestrzeni ryzyko–zysk.

```
from amplpy import AMPL, add_to_path
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
AMPL_MODEL = "main2.mod"
AMPL_DATA = "main2.dat"
add_to_path(r"E:\AMPL\ampl_mswin64")
ampl = AMPL()
ampl.read(AMPL_MODEL)
ampl.readData(AMPL_DATA)
ampl.setOption('solver', 'cplex')
ampl.setOption("solver_msg", 0)
ampl.getParameter("w_profit").set(1)
ampl.getParameter("w_risk").set(0)
ampl.solve()
max_profit = ampl.getVariable("ExpProfit").value()
ampl.getParameter("w_profit").set(1)
ampl.getParameter("w_risk").set(200)
ampl.solve()
min_profit = ampl.getVariable("ExpProfit").value()
ampl.eval("""
param min_profit;
subject to MinProfitConstraint:
    ExpProfit >= min_profit;

# Zmiana funkcji celu na minimalizację ryzyka
maximize MinRiskOnly:
    -Risk;
""")
profit_values = np.linspace(min_profit, max_profit, 21)[::-1]
results = []
for min_profit_val in profit_values:
    ampl.getParameter("min_profit").set(min_profit_val)
    ampl.solve()
    exp_profit = ampl.getVariable("ExpProfit").value()
    risk = ampl.getVariable("Risk").value()
    results.append({"MinProfit": min_profit_val, "ExpProfit": exp_profit, "Risk": risk})
    print(f"Min Profit: {min_profit_val:.2f}, Osiągnięty zysk: {exp_profit:.2f}, Ryzyko: {risk:.2f}")
df = pd.DataFrame(results)
df.to_csv("pareto_results3.csv", index=False)
df_sorted = df.sort_values(by="Risk")
plt.figure(figsize=(8, 6))
plt.plot(df_sorted["Risk"], df_sorted["ExpProfit"],
         marker='o', color='blue', label='Rozwiązania Pareto')
plt.xlabel("Ryzyko")
plt.ylabel("Oczekiwany zysk")
plt.title("Obraz zbioru rozwiązań efektywnych (Pareto)")
plt.legend()
plt.grid(True)
plt.show()
```

Rys. 6. Kod w Pythonie odpowiedzialny za sprawdzenie scenariuszy dla różnych wag ryzyka



Rys. 7. Obraz zbioru rozwiązań efektywnych w przestrzeni ryzyko-zysk z równomiernym próbkowaniem

W celu wykonania powyższego zmodyfikowałem kod AMPL z początku tego zadania zmieniając funkcję celu z maksymalizacji ważonego ryzyka i zysku na minimalizację ryzyka przy oczekiwanym poziomie zysku, gdyż średnia ważona nie da nam możliwości osiągnięcia pożądanej ilości wyników i rozwiązań Pareto:

```
param min_profit;

subject to MinProfitConstraint:
    ExpProfit >= min_profit;

# Zmiana funkcji celu na minimalizację ryzyka
minimize MinRiskOnly:
    Risk;
```

Rys. 8. Część kodu odpowiedzialna za implementację awersji do ryzyka przy określonym zysku.

Zatem zmiana pozwoliła na określonym poziomie oczekiwanego zysku wybrać odpowiednie rozwiązanie w kontekście ryzyka, co z kolei pozwoliło na odpowiednie przedstawienie obszaru rozwiązań efektywnych z tego punktu zadania.

b. Wskazać rozwiązania efektywne minimalnego ryzyka i maksymalnego zysku. Jakie odpowiadają im wartości w przestrzeni ryzyko–zysk?

W tym przypadku rozwiązaniem efektywnym minimalnego ryzyka jest rozwiązanie najbliższe początku układu współrzędnych, odpowiada mu zysk 99900,00zł i ryzyko 0,00zł. Rozwiązaniem efektywnym maksymalnego zysku jest 123358,08zł z ryzykiem 425,11zł.

c. Wybrać trzy dowolne rozwiązania efektywne. Sprawdzić czy zachodzi pomiędzy nimi relacja dominacji stochastycznej pierwszego rzędu. Wyniki skomentować, odnieść do ogólnego przypadku.

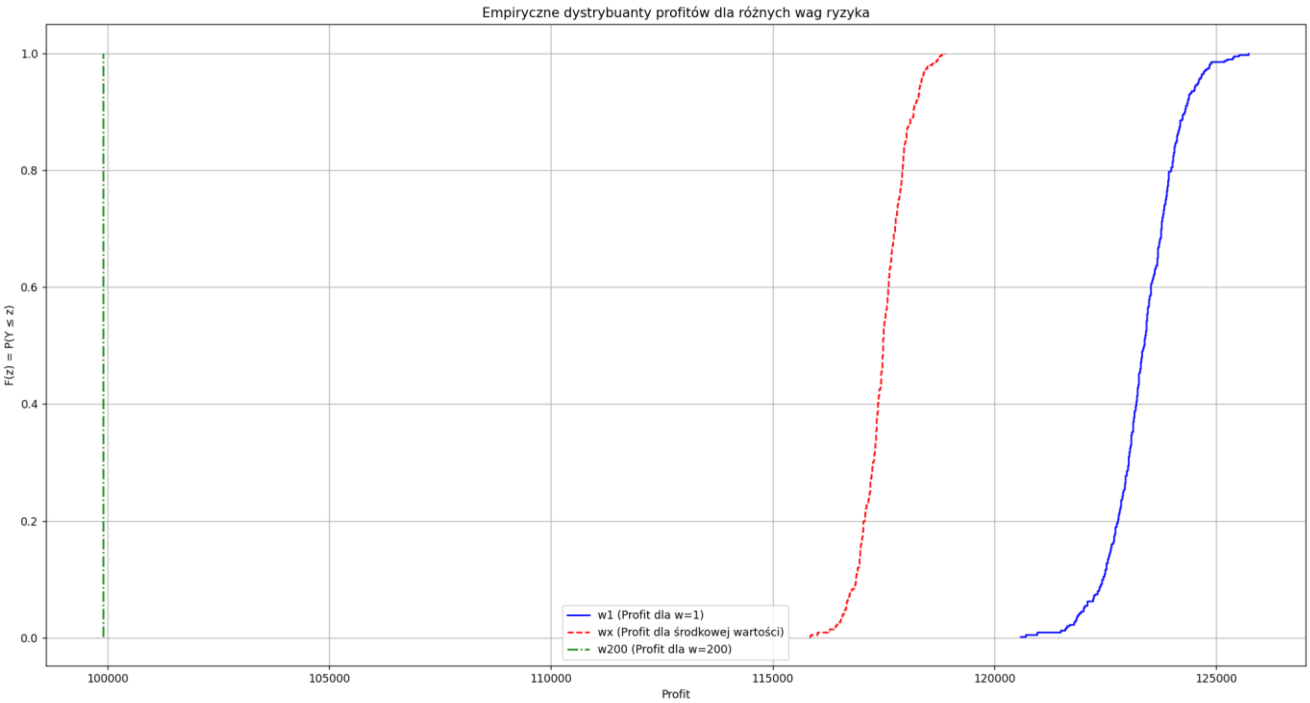
Za rozwiązania wybieram rozwiązanie efektywne minimalnego ryzyka, maksymalnego zysku oraz losowe ze „środka”, zatem zysk 117493,56zł i ryzyko 287,07zł. Rozwiązanie A dominuje stochastycznie w pierwszym rzędzie rozwiązanie B wtedy i tylko wtedy, gdy dla każdej wartości z zachodzi: $F_A(z) \leq F_B(z)$, przy czym dla co najmniej jednej wartości z musi zachodzić nierówność ostra. W celu sprawdzenia stochastycznej dominacji potrzebuję zatem obliczenia dystrybuant, gdzie w ogólnym przypadku dystrybuanta rozwiązania dominującego będzie leżała na prawo lub pokrywała się z dystrybuantą rozwiązania zdominowanego.

Logika kodu w Pythonie odpowiedzialnego za narysowanie wykresów dystrybuanty:

```
probability = 1.0 / n_scenarios
p_array = np.full(n_scenarios, probability)


Tabnine | Edit | Test | Explain | Document
def compute_cdf(profits):
    sorted_indices = np.argsort(profits)
    sorted_profits = profits[sorted_indices]
    cdf = np.cumsum(p_array[sorted_indices])
    return sorted_profits, cdf


profits_w1_sorted, cdf_w1 = compute_cdf(profits_w1)
profits_w40_sorted, cdf_w40 = compute_cdf(profits_w40)
profits_w200_sorted, cdf_w200 = compute_cdf(profits_w200)

plt.figure(figsize=(8, 6))
plt.step(profits_w1_sorted, cdf_w1, where='post', label='w1 (Profit dla w=1)', linestyle='-', color='b')
plt.step(profits_w40_sorted, cdf_w40, where='post', label='w40 (Profit dla w=40)', linestyle='--', color='r')
plt.step(profits_w200_sorted, cdf_w200, where='post', label='w200 (Profit dla w=200)', linestyle='-.', color='g')

plt.xlabel('Profit')
plt.ylabel('F(z) = P(Y ≤ z)')
plt.title('Empiryczne dystrybuanty profitów dla różnych wag ryzyka')
plt.legend()
plt.grid(True)

plt.show()
```

Rys. 9. Kod w Pythonie do rysowania wykresów dystrybuanty

W dwukryterialnym modelu zysk-ryzyko rozwiązania o niższej wadze ryzyka zapewniają wyższe oczekiwane zyski kosztem wyższego ryzyka. Natomiast rozwiązania o wyższej wadze ryzyka oferują niższe ryzyko, ale również niższe oczekiwane zyski. W kontekście ogólnym, nie zawsze musi zachodzić relacja dominacji stochastycznej pierwszego rzędu między rozwiązaniami efektywnymi, ale w tym przypadku jej występowanie świadczy o silnym uporządkowaniu rozwiązań według preferencji decydenta względem ryzyka.



Rys. 10. Wykres dystrybuant zysków dla różnych wag ryzyka

Porównanie w200 i wx oraz w1: Dystrybuanta dla w200 (zielona przerywana z kropkami) leży całkowicie na lewo od dystrybuanty dla wx (czerwona przerywana) oraz w1 (niebieska ciągła). Oznacza to, że dla każdej wartości zysku z, prawdopodobieństwo osiągnięcia zysku mniejszego lub równego z jest większe dla rozwiązania w200 niż dla wx lub w1. Innymi słowy, rozwiązania wx i w1 dominują stochastycznie w pierwszym rzędzie rozwiązanie w200.

Porównanie w1 i wx: Podobnie, dystrybuanta dla wx (czerwona przerywana) leży prawie całkowicie na lewo od dystrybuanty dla w1 (niebieska ciągła). Oznacza to, że rozwiązanie w1 dominuje stochastycznie w pierwszym rzędzie rozwiązanie wx.

Przedstawiona tutaj miara ryzyka nie jest zgodna z aksjomatami dominacji stochastycznej pierwszego rzędu, gdyż występują w niej jedynie rozwiązania zdominowane przez rozwiązanie maksymalnego zysku. Nie mamy za bardzo wyboru jako logicznie działający interesant, gdyż zawsze powinniśmy wybrać to samo rozwiązanie.

Zadanie dodatkowe

Przetestowałem modyfikację miary ryzyka Giniego, przekształcając ją zgodnie z dostarczonymi materiałami w miarę bezpieczeństwa opisaną w pełni w rozdziale:

1.4. Średnia różnica Giniego

Miara:

$$G(\mathbf{y}) = \frac{1}{2} \sum_{t'=1}^{T} \sum_{t''=1}^{T} |y_{t'} - y_{t''}| p_{t'} p_{t''}$$

Twierdzenie: dla dowolnego $0 < \lambda \leqslant 1$ rozwiązanie optymalne zadania

$$\max \{\mu(\mathbf{f}(\mathbf{x})) - \lambda G(\mathbf{f}(\mathbf{x})) \ : \ \mathbf{x} \in Q\}$$

jest rozwiązaniem SSD efektywnym.

Model optymalizacyjny:

$$\begin{aligned}
\max \quad & \sum_{t=1}^{T} p_t y_t - \lambda \sum_{t'=1}^{T} \sum_{t'' \neq t'} d_{t't''} p_{t'} p_{t''} \\
\text{p.w.} \quad & d_{t't''} \geqslant y_{t'} - y_{t''}, \ d_{t't''} \geqslant 0 \quad \text{dla } t', t'' = 1, \ldots, T; \ t'' \neq t' \\
& y_t = f_t(\mathbf{x}) \quad \text{dla } t = 1, \ldots, T \\
& \mathbf{x} \in Q
\end{aligned}$$

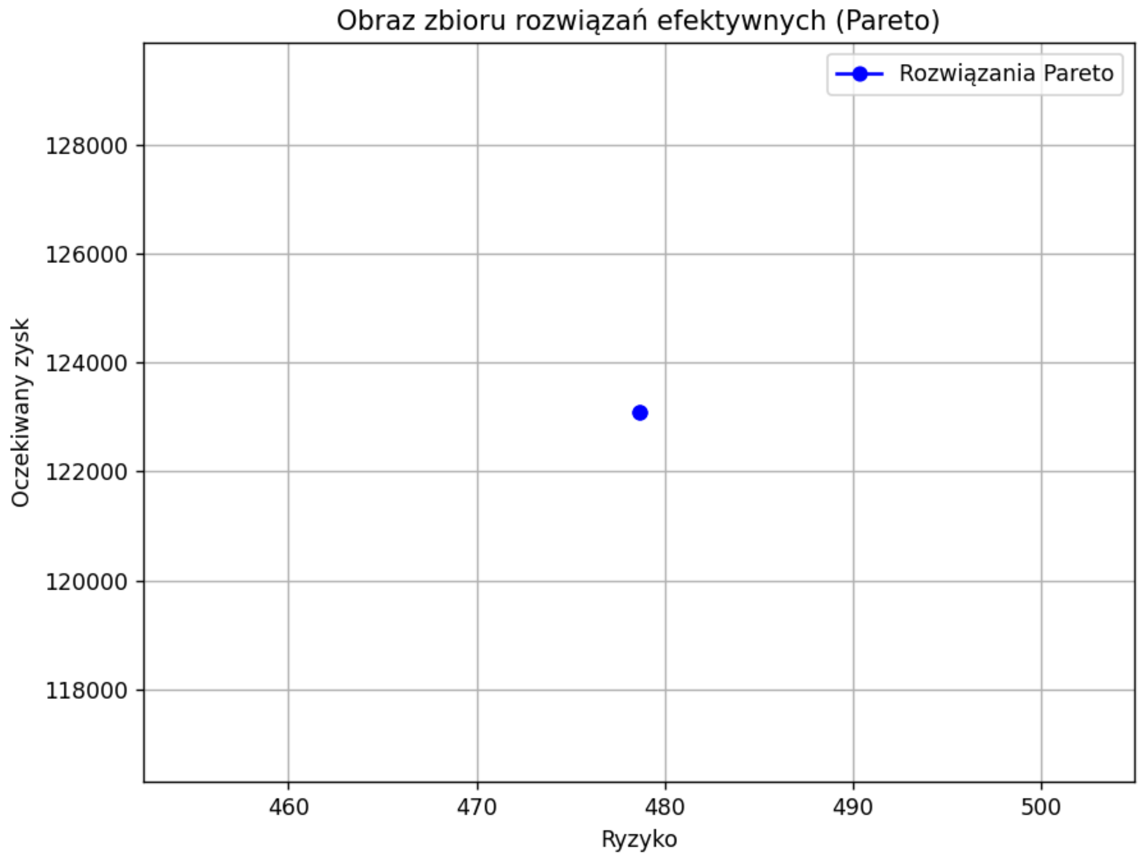
Rys. 11. Definicja miary bezpieczeństwa Giniego

W ten sposób nie będą generowane rozwiązania które nie są efektywne względem FSD.

```
# Metoda ważona
param lambda >= 0;
param min_profit;

subject to MinProfitConstraint:
    ExpProfit >= min_profit;

var security = ExpProfit - lambda * Risk;

# Funkcja celu - chcemy maksymalizować "Security" - miarę bezpieczeństwa
maximize Security:
   security;
```

Rys. 12. Edytowana część kodu dostosowująca do miary bezpieczeństwa

Przekształcając w ten sposób kod udało mi się uzyskać następujący obszar zbioru rozwiązań efektywnych:



Rys. 13. Obraz zbioru rozwiązań efektywnych w przestrzeni ryzyko-zysk z równomiernym próbkowaniem dla miary bezpieczeństwa

Jak można zauważyć na powyższym wykresie – nie zostają wygenerowane inne rozwiązania niż rozwiązanie maksymalnego zysku. Odpowiednikiem ryzyka z tego wykresu może być miara bezpieczeństwa która wynosi tutaj 122615,37. Potwierdza to zatem, że nie zostają przez miarę bezpieczeństwa generowane rozwiązane inne niż rozwiązanie optymalne w kontekście FSD. Aby otrzymać bardziej obrazowe wyniki zdecydowałem się zwiększyć wartości „d" dziesięciokrotnie.

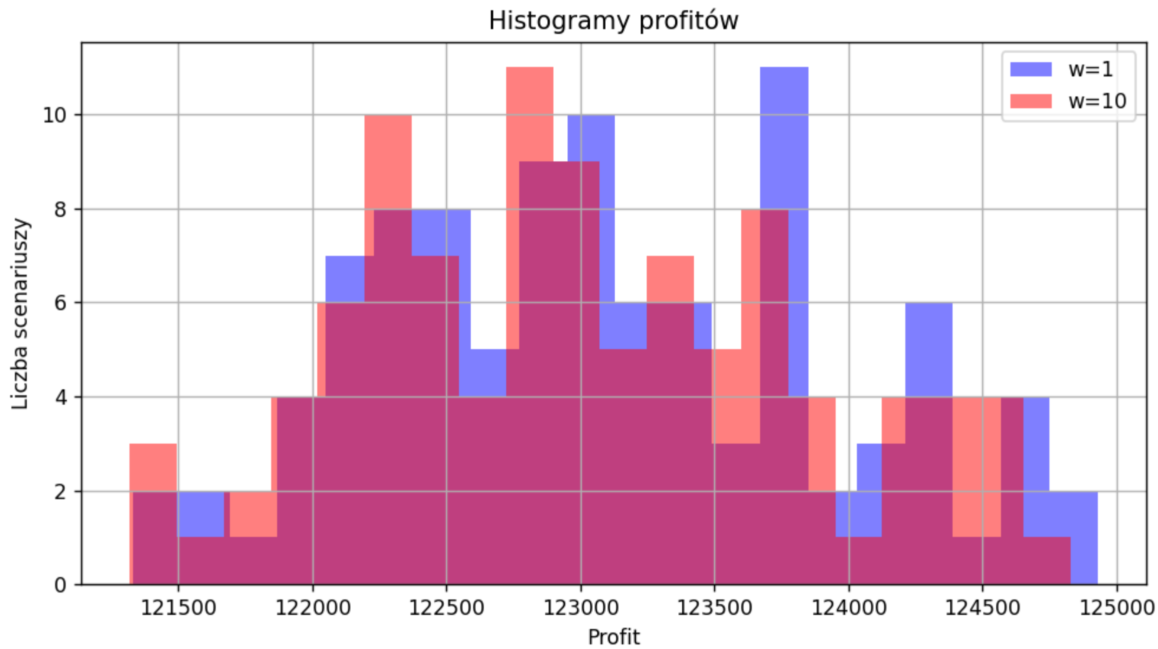
Rys. 14. Obraz zbioru rozwiązań efektywnych w przestrzeni ryzyko-zysk z równomiernym próbkowaniem dla miary bezpieczeństwa z dziesięciokrotnie większym współczynnikiem D

Zauważyć można teraz że zostały wygenerowane dwa rozwiązania efektywne które w kontekście dominacji stochastycznej oczywiście że nie będą już poprawne, gdyż dokonałem tutaj ręcznego mnożenia, jednak pozwalają one na dalszą analizę:

Porównanie statystyk profitów dla różnych wag ryzyka

	w=1	w=10	Różnica %
Średnia	123094.0	123014.3	-0.06
Mediana	123085.0	122989.0	-0.08
Odchylenie std.	838.46	822.11	-1.95
Minimum	121330.0	121318.0	-0.01
Maksimum	124930.0	124827.0	-0.08
Zakres	3600.0	3509.0	-2.53

Rys. 15. Porównanie statystyk dla różnych wag ryzyka



Rys. 16. Porównanie histogramów dla różnych wag ryzyka

Analiza powyższych danych pozwala stwierdzić, że nie otrzymujemy innych rozwiązań w przypadku naszego zadania z uwagi na wyjątkowo niską skalę zmian w zakresie oczekiwanych zysków co z kolei rzutuje na bardzo niskie wartości ryzyka, oraz praktyczny ich brak wpływu na miarę bezpieczeństwa. Wskazuje na to przede wszystkim wartość odchylenia standardowego i różnicy między najlepszym i najgorszym scenariuszem. Gdybyśmy znacząco rozszerzyli histogram (zwiększyli odległość między najlepszym i najgorszym scenariuszem), miara bezpieczeństwa przedstawiona w tym zadaniu prawdopodobnie byłaby w stanie zaprezentować co najmniej dwa niezdominowane w sensie FSD rezultaty.