中附中１年生
理工学部訪問

中央大学：髙木悠人

議題

- 自己紹介
- 学科の紹介
- 理系の魅力
- 研究の楽しさ・魅力

自己紹介

名前: **髙木悠人(たかぎ　ゆうと)**

学科: 理工学部**ビジネスデータサイエンス**学科

分野: AI, データサイエンス(大規模なデータの解析等), 統計学

なぜ理系に進んだ??

- **AIの仕組み**に興味があったから
- 横文字がカッコよかったから
- 専門的な強みをつけたかったから

ビジネスデータサイエンス学科の魅力は??

- 世界の最先端(AI, データサイエンス)を知れること

ビジネスデータサイエンス学科

理工学部
(~26年)

基幹
理工学部

社会
理工学部

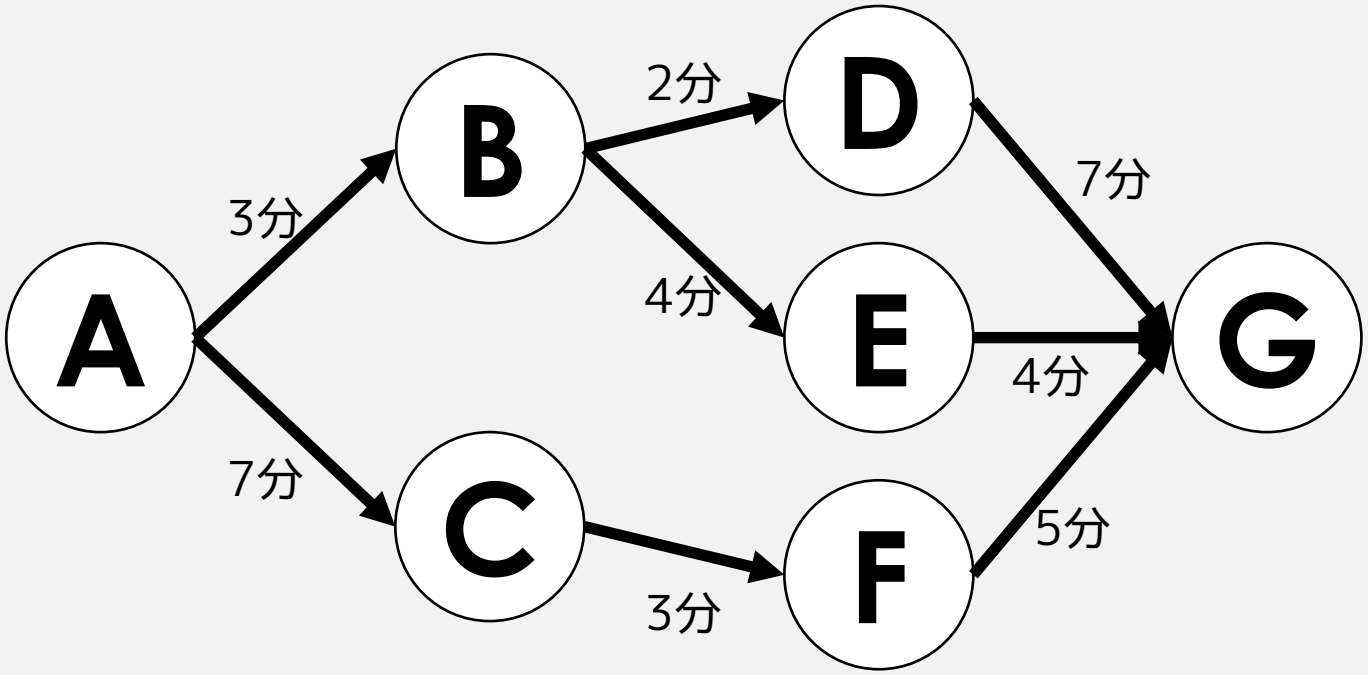
先進
理工学部

- 数学科
- 物理学科
- 応用化学科
- 生命科学科
- 都市環境学科
- ビジネスデータサイエンス学科
- 人間総合理工学科
- 精密機械工学科
- 電気電子情報通信工学科
- 情報工学科

ビジネスデータサイエンス学科

どんなことやってるの？？

Q. **AからGまで最短時間**で行くときのルートは？？

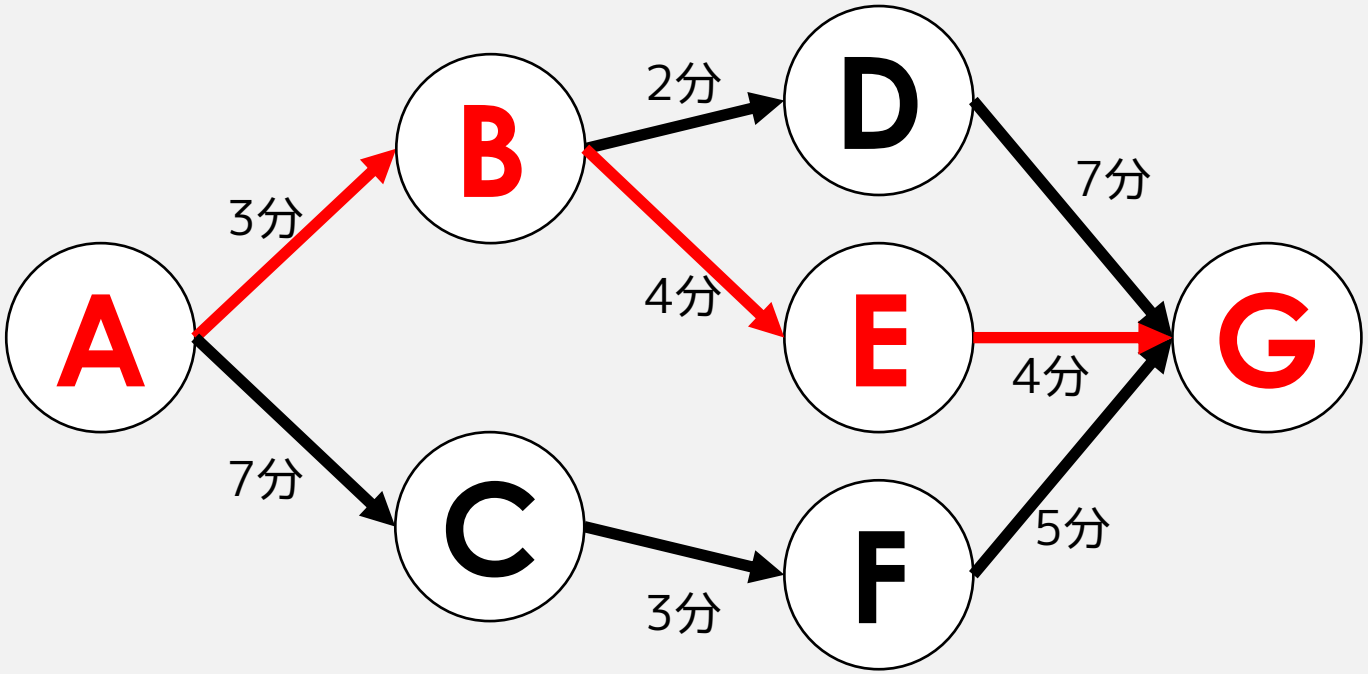


参考: OR(オペレーションズリサーチ第一)

ビジネスデータサイエンス学科

どんなことやってるの？？

Q. **AからGまで最短時間**で行くときのルートは？？



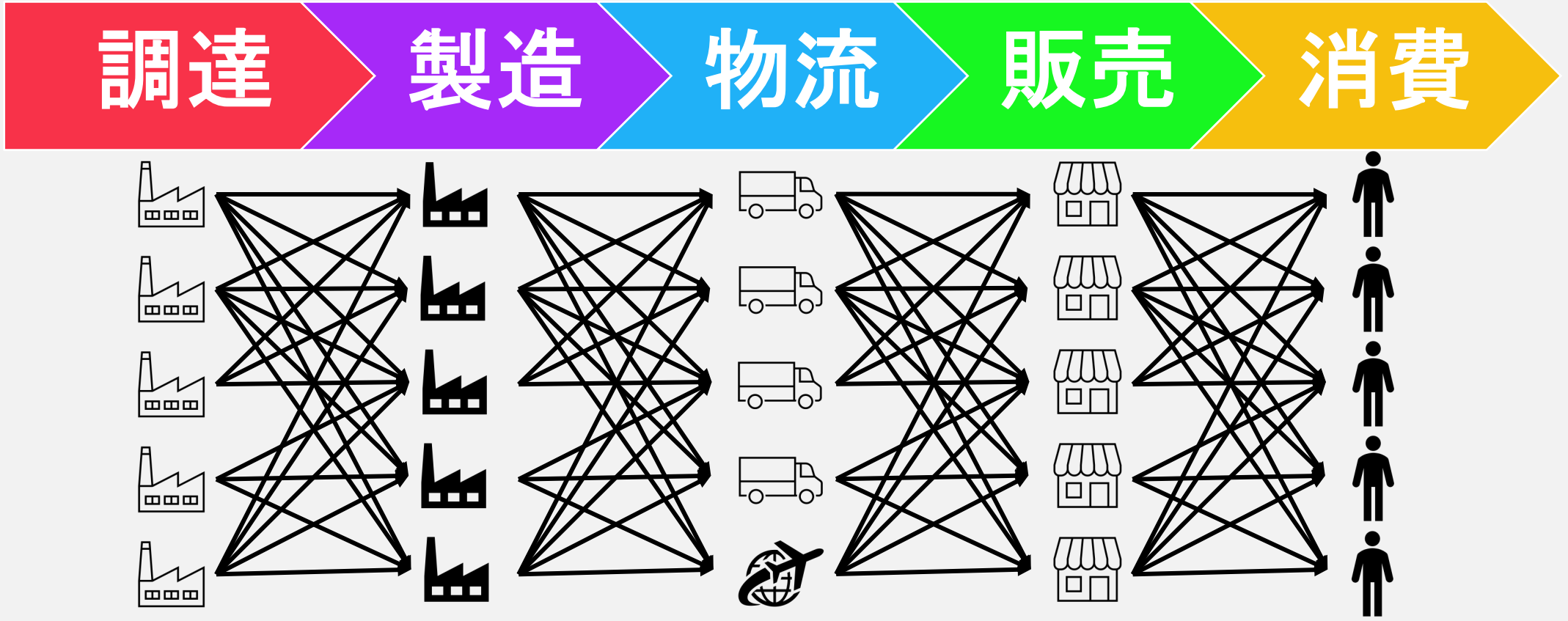
合計: 11分

参考: OR(オペレーションズリサーチ第一)

ビジネスデータサイエンス学科

大学では、複雑なバージョンを解いている。

例. 材料調達から消費までの輸送最安ルートの検討



ビジネスデータサイエンス学科

大学では、複雑なバージョンを解いている。

```python
# -*- coding: utf-8 -*-
"""
SCMを線形計画問題（最小費用フロー）として解くサンプル
レイヤ: Suppliers -> Plants -> DCs -> Stores -> Customers
変数: 各レイヤ間のフロー量
目的: 総輸送費（＋任意で製造・保管コスト）最小化
制約: 供給上限、節点フロー保存、各施設容量、需要充足
"""

import pulp as pl

# ------------------------------
# 1) 入力データ(例)
# ------------------------------
Suppliers = ["S1", "S2", "S3"]
Plants    = ["P1", "P2"]
DCs       = ["D1", "D2"]
Stores    = ["R1", "R2", "R3"]   # Retail
Customers = ["C1", "C2", "C3", "C4"]

# 供給量(サプライヤの出荷上限)
supply = {
    "S1": 80,
    "S2": 60,
    "S3": 50,
}

# 需要量(顧客の需要)
demand = {
    "C1": 40,
    "C2": 55,
    "C3": 50,
    "C4": 20,
}

# 各施設の処理容量(通過量の上限)
plant_capacity = {"P1": 120, "P2": 100}
dc_capacity    = {"D1": 110, "D2": 90}
store_capacity = {"R1": 80, "R2": 90, "R3": 80}

# コスト(単位流量あたり)
# ない経路は辞書に載せない=禁止アーク
cost_SP = {  # Supplier -> Plant
    ("S1","P1"): 2, ("S1","P2"): 6,
    ("S2","P1"): 4, ("S2","P2"): 3,
    ("S3","P1"): 5, ("S3","P2"): 4,
}
```

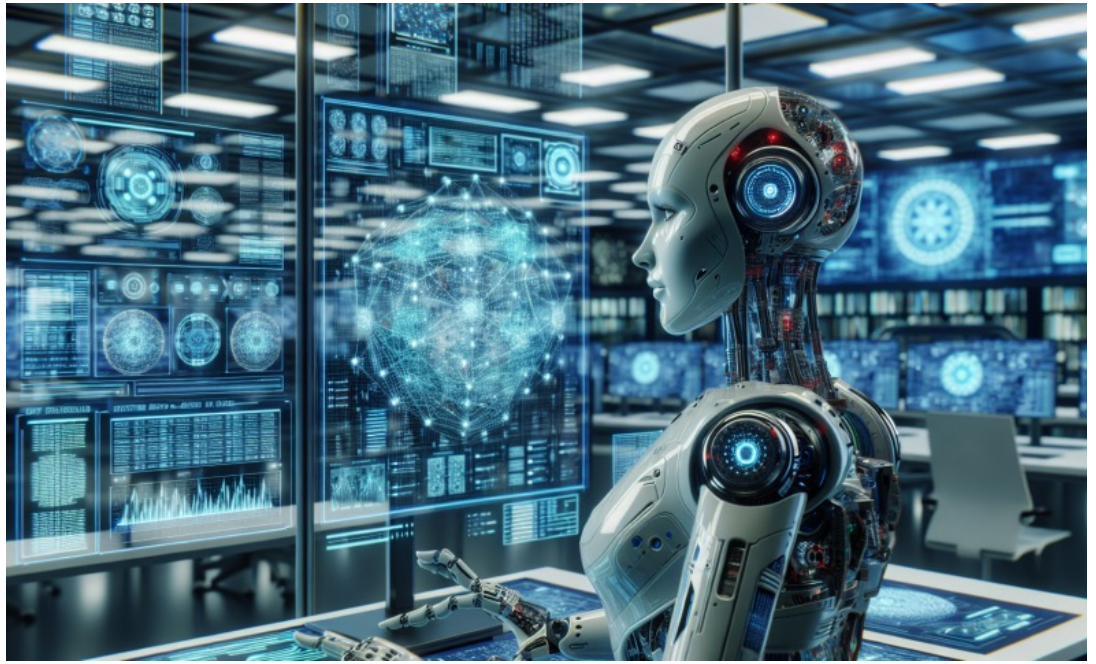
```python
cost_PD = {  # Plant -> DC
    ("P1","D1"): 3, ("P1","D2"): 4,
    ("P2","D1"): 2, ("P2","D2"): 5,
}
cost_DR = {  # DC -> Store
    ("D1","R1"): 2, ("D1","R2"): 6, ("D1","R3"): 4,
    ("D2","R1"): 3, ("D2","R2"): 2, ("D2","R3"): 5,
}
cost_RC = {  # Store -> Customer
    ("R1","C1"): 3, ("R1","C2"): 2, ("R1","C3"): 6, ("R1","C4"): 4,
    ("R2","C1"): 4, ("R2","C2"): 3, ("R2","C3"): 2, ("R2","C4"): 5,
    ("R3","C1"): 5, ("R3","C2"): 4, ("R3","C3"): 3, ("R3","C4"): 2,
}


# ------------------------------
# 2) モデル定義
# ------------------------------
model = pl.LpProblem("SCM_MinCostFlow", pl.LpStatusOptimal)

# 変数:各レイヤ間のフロー(非負連続)
x_SP = pl.LpVariable.dicts("x_SP", cost_SP.keys(), lowBound=0, cat="Continuous")
x_PD = pl.LpVariable.dicts("x_PD", cost_PD.keys(), lowBound=0, cat="Continuous")
x_DR = pl.LpVariable.dicts("x_DR", cost_DR.keys(), lowBound=0, cat="Continuous")
x_RC = pl.LpVariable.dicts("x_RC", cost_RC.keys(), lowBound=0, cat="Continuous")

# 目的関数:総コスト最小化
model += (
    pl.lpSum(cost_SP[a]*x_SP[a] for a in cost_SP) +
    pl.lpSum(cost_PD[a]*x_PD[a] for a in cost_PD) +
    pl.lpSum(cost_DR[a]*x_DR[a] for a in cost_DR) +
    pl.lpSum(cost_RC[a]*x_RC[a] for a in cost_RC)
), "TotalCost"

# ------------------------------
# 3) 制約
# ------------------------------

# (A) 供給上限:各サプライヤの出荷 ≤ 供給量
for s in Suppliers:
    outflow = pl.lpSum(x_SP[(s,p)] for p in Plants if (s,p) in x_SP)
    model += outflow <= supply.get(s, 0), f"SupplierCap_{s}"

```

```python

# (B) フロー保存:各Plantで入出一致、かつ容量上限
for p in Plants:
    inflow  = pl.lpSum(x_SP[(s,p)] for s in Suppliers if (s,p) in x_SP)
    outflow = pl.lpSum(x_PD[(p,d)] for d in DCs      if (p,d) in x_PD)
    model += inflow == outflow, f"FlowBalance_P_{p}"
    model += outflow <= plant_capacity.get(p, 0), f"PlantCap_{p}"

# (C) フロー保存:各DCで入出一致、かつ容量上限
for d in DCs:
    inflow  = pl.lpSum(x_PD[(p,d)] for p in Plants if (p,d) in x_PD)
    outflow = pl.lpSum(x_DR[(d,r)] for r in Stores if (d,r) in x_DR)
    model += inflow == outflow, f"FlowBalance_D_{d}"
    model += outflow <= dc_capacity.get(d, 0), f"DcCap_{d}"

# (D) フロー保存:各店舗で入出一致、かつ店舗容量
for r in Stores:
    inflow  = pl.lpSum(x_DR[(d,r)] for d in DCs if (d,r) in x_DR)
    outflow = pl.lpSum(x_RC[(r,c)] for c in Customers if (r,c) in x_RC)
    model += inflow == outflow, f"FlowBalance_R_{r}"
    model += outflow <= store_capacity.get(r, 0), f"StoreCap_{r}"

# (E) 需要充足:各顧客には需要量ちょうどを配達
for c in Customers:
    inflow = pl.lpSum(x_RC[(r,c)] for r in Stores if (r,c) in x_RC)
    model += inflow == demand.get(c, 0), f"Demand_{c}"

# (任意)総供給量 ≥ 総需要量の可行性チェック(不足時は不可解)
total_supply = sum(supply.values())
total_demand = sum(demand.values())
assert total_supply >= total_demand, "総供給量が総需要量を下回っています。データを見直してください。"

# ------------------------------
# 4) 求解
# ------------------------------
status = model.solve(pl.PULP_CBC_CMD(msg=False))
print("Status:", pl.LpStatus[status])
print("Optimal Cost:", pl.value(model.objective))

# ------------------------------
# 5) 解の出力
# ------------------------------
def print_positive(flows, title):
    print(f"\n[{title}]")
    for k, var in flows.items():
        if var.varValue is not None and var.varValue > 1e-6:
            print(f"{k}: {var.varValue:.2f}")

print_positive(x_SP, "Supplier -> Plant")
print_positive(x_PD, "Plant -> DC")
print_positive(x_DR, "DC -> Store")
print_positive(x_RC, "Store -> Customer")
```

ビジネスデータサイエンス学科

Q. みなさん、どのAI使ってますか??

1. ChatGPT
2. Gemini
3. Claude
4. 他のツール



僕は、ChatGPTとClaude使ってます！

ビジネスデータサイエンス学科

Q. AIがどうやって動いてるかわかりますか？？

「AIが文章の意味を理解する」ってどういう仕組み？

ビジネスデータサイエンス学科

実は、AIは文章を理解しているのではなく、確率計算を無限にやっている！！

参考:スキルアップAI Journal

理系の魅力

魅力1. 将来活かせる専門性が身に付く

魅力2. 問題解決をする上での論理的思考力・ツールを身につけられる

理系の魅力

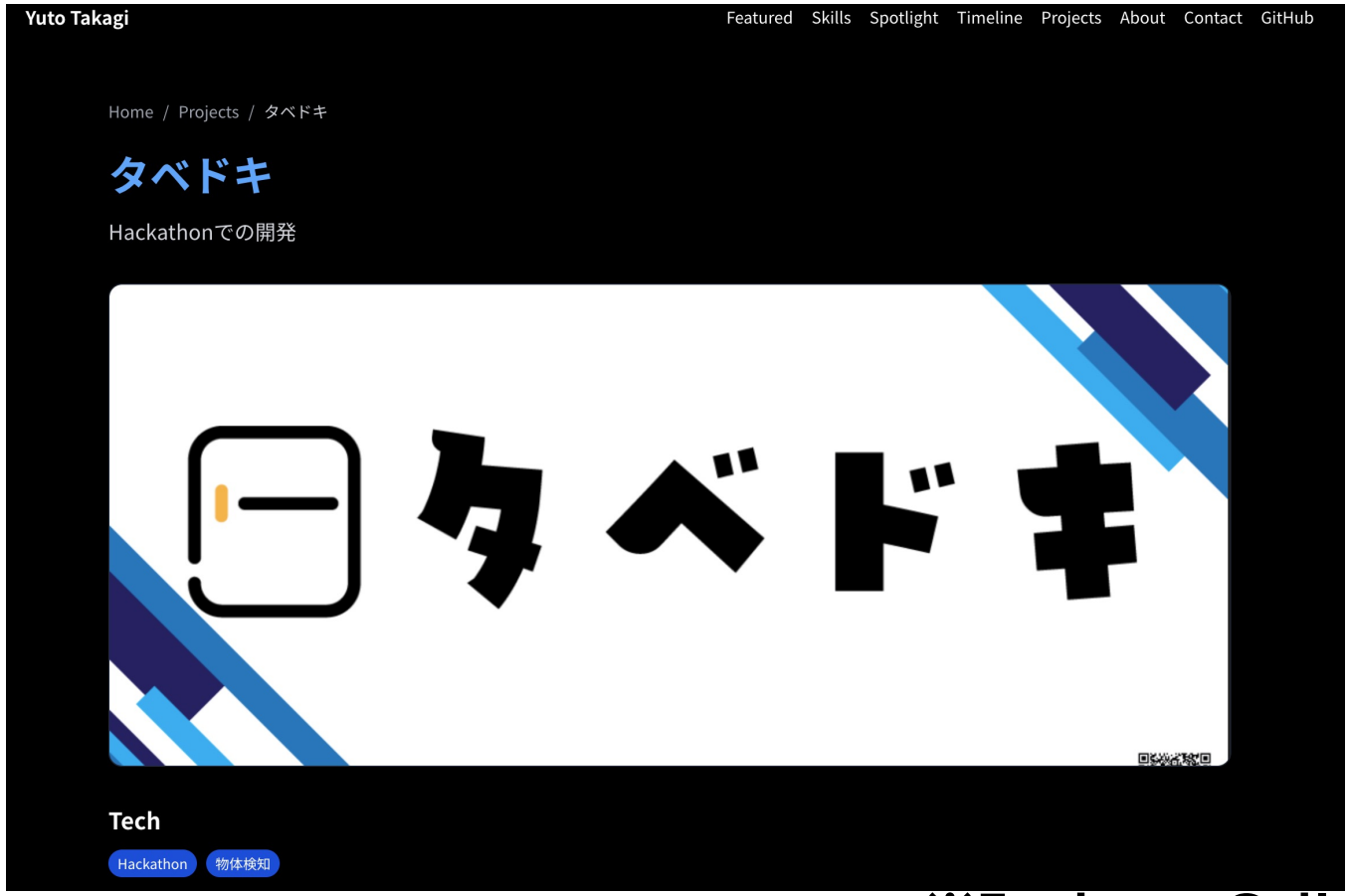
魅力1. 将来活かせる専門性が身に付く

大学(学科)で勉強すること	活かせる職業
OR 輸送ルートの最適化など	**SAPコンサルティング,** **AIコンサルティング**
プログラミング 主に、AI, 機械学習系	**AIエンジニア, MLエンジニア,** **AIコンサルティング**
※アントレプレナーシップ ビジネスアイデアを形にする授業	**起業, アドバイザリー**

理系の魅力

魅力2. 問題解決をする上での論理的思考力・ツールを身につけられる

例) 冷蔵庫の中の写真を撮って、作れるメニュー提案してくれるアプリ

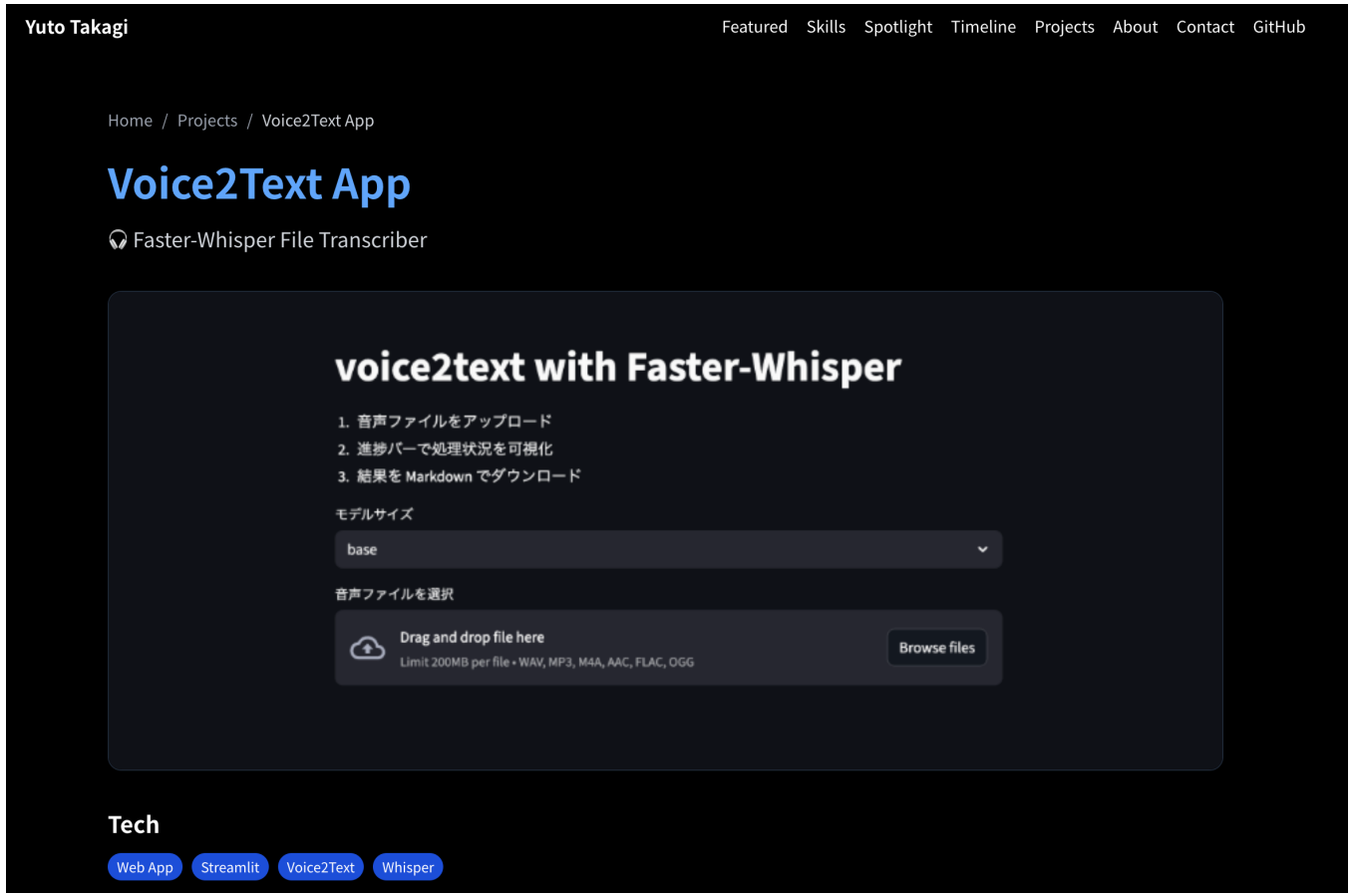


※Engineer Guild Hackathonにて、開発

理系の魅力

魅力2. 問題解決をする上での論理的思考力・ツールを身につけられる

例) 会話文字起こしアプリ



理系の魅力

魅力2. 問題解決をする上での論理的思考力・ツールを身につけられる

例) オンライン ケース面接プラットフォーム(G-SPARKの一貫で開発中)

事業アイデアの実現を目指す年間プログラム「G-SPARK⚡」

2025年04月01日　#グローバルアントレプレナーシップ　#アントレプレナーシップ　#起業家　#起業家精神　#G-SPARK

研究の楽しさ・魅力

研究内容: 静止摩擦力における待機時間依存性・接触面積依存性の研究

簡単に言うと..

高校物理では、「摩擦力はモノの重さに比例する」と言われている

↕

置き続ければ(待機時間)、摩擦大きくなりそう！

面積大きければ(接触面積)、摩擦大きくなりそう！

研究の楽しさ=「ギモンに思ったこと」を解決できること

ご清聴ありがとうございました

YutoTAKAGI
a23.nhm4@g.chuo-u.ac.jp
https://portfolio-ruw9.onrender.com/