卒業研究進捗報告

nanimonaiyo

山本 藤也（Touya Yamamoto）
u220067@st.pu-toyama.ac.jp

富山県立大学 情報システム工学科

June 24, 2025

TOYAMA Prefectural University

前回の内容

目的

- 学生の履修選択を支援するシステムの開発
- 過去の先輩の就職先・成績データに基づく履修指標提示機能の追加を目指す

アプローチ

- 滝沢先輩のプログラムを修正し、要素を追加する

やったこと１

問題

- 科目や講義回ごとに Flask のルート関数が個別に存在した。
  （例: /ComputerNetWork_0, /ComputerNetWork_1 ...）
  その結果、科目や講義の数に比例してコード量が爆発的に増加。
- 表示レイアウトを少し変更するだけでも、数十箇所のファイルを修正する必要があり、非効率かつ修正漏れを誘発しやすい構造だった。

```
1080  #------------コンピュータネットワーク----------------------------
1081  #------------根本のhtml---------------------------------
1082  @app.route("/ComputerNetWork")
1083  def ComputerNetWork():
1084      return render_template("/ComputerNetWork/ComputerNetWork.html")
1085  #-----------------------------------------------------
1086  #-----------------第一回-------------------------------
1087  @app.route('/ComputerNetWork_0')
1088  def ComputerNetWork_0():
1089      kyouzaiDB = pd.read_csv('app/Kamoku/ComputerNetWork/Kyouzai/kyouzai_0.csv')
1090      kyouzai_title_list = []
1091      kyouzai_url_list = []
1092      mihyouka_kyouzai_title_list = []
1093      mihyouka_kyouzai_url_list = []
1094      for i in range(len(kyouzaiDB.index)):
1095          kyouzai_title_list.append(kyouzaiDB.iloc[i,0])
1096          kyouzai_url_list.append(kyouzaiDB.iloc[i,1])
1097      mihyouka_kyouzai_title_list.append(kyouzaiDB.iloc[-1]['HPtitle'])
1098      mihyouka_kyouzai_url_list.append(kyouzaiDB.iloc[-1]['HPurl'])
1099      mihyouka_kyouzai_title_list.append(kyouzaiDB.iloc[-2]['HPtitle'])
1100      mihyouka_kyouzai_url_list.append(kyouzaiDB.iloc[-2]['HPurl'])
```

TOYAMA Prefectural University

やったこと1

解決策

- 「動的ルート」の導入 (/kamoku/科目名/lecture/講義 ID)
- データ取得やレビュー処理のロジックを「共通ヘルパー関数」に集約。

効果

- 数十個あった関数を、わずか 3 つの汎用的な関数に削減。
- 新しい科目の追加が、Python コードを一切変更せずにフォルダとデータファイルの配置だけで可能になった。
  → 拡張性と保守性が劇的に向上した。

やったこと2

問題

- 科目情報（科目名、フォルダ名、区分、単位数など）が、app.py 内の巨大な辞書と、単一のデータベーステーブルに分散・固定（ハードコード）されていた。
- この構造では、「2019 年度生と 2020 年度生で、同じ科目名でも単位数や区分が違う」といった、現実のカリキュラム変更に対応できなかった。

```
263         suisenn_all_url = []
264         suisenn_all_name  = []
265         for i in range(len(suisenn_all)):
266             if suisenn_all[i] == ('インターネット工学'):
267                 suisenn_all_url.append('intanettokougaku')
268                 suisenn_all_name.append('インターネット工学')
269             elif suisenn_all[i] == ('経済学Ⅰ'):
270                 suisenn_all_url.append('keizaigaku1')
271                 suisenn_all_name.append('経済学Ⅰ')
272             elif suisenn_all[i] == ('経済学Ⅱ'):
273                 suisenn_all_url.append('keizaigaku2')
274                 suisenn_all_name.append('経済学Ⅱ')
275             elif suisenn_all[i] == ('環境論Ⅰ'):
276                 suisenn_all_url.append('kannkyouronn1')
277                 suisenn_all_name.append('環境論Ⅰ')
278             elif suisenn_all[i] == ('社会学Ⅰ'):
279                 suisenn_all_url.append('syakaigaku1')
280                 suisenn_all_name.append('社会学Ⅰ')
281             elif suisenn_all[i] == ('日本語表現法'):
282                 suisenn_all_url.append('nihonngohyougennhou')
283                 suisenn_all_name.append('日本語表現法')
284             elif suisenn_all[i] == ('芸術学'):
```

やったこと2

解決策

- 「年度別マスター CSV」 (catalogs/course_catalog_YYYY.csv) を唯一の情報源とする、新しいデータ管理アーキテクチャを設計・導入。
- ログインした学生の年度に合わせ、対応する年度のマスター CSV を動的に読み込む仕組みに変更。

効果

- 学年ごとに異なるカリキュラムに完全に対応可能になった。
- 科目に関する全ての情報がデータファイルに一元化され、コードから分離された。
  → 将来の仕様変更に強い、柔軟なシステムを実現した。

やったこと３

問題

- 教員用のシラバス作成ページを組み込めたが、以前までのプログラムでは保存先も数学１に固定されており、作成されたシラバスを表示する機能も実装されていなかった。

```
286
287     ######################################################################
288
289     # こっちはhtmlで保存するようにする
290     # 入力された状態のhtmlを html にいれる.
291     html = render_template('true_syllabus.html', sub_name=kamoku_name,  teacher=tantou_name, kamoku_eng=kamoku_eng
292                                             educational_goal=educational_goal,outline=outline, student_goal=studen
293
294     os.makedirs("html_save", exist_ok=True)
295     # 上で作成した html を aiueo.html という名前で保存する. (このaiueo.htmlを科目名に変更するかどうか)
296     # with open("html_save/aiueo.html",mode="w",encoding="utf-8") as f:
297     with open("templates/数学１/aiueo.html", mode="w", encoding="utf-8") as f:
298         f.write(str(html))
299
300     return render_template('teacher.html', tantou_list = kyouin)
301
302
303
```

やったこと3

解決策

- 科目と同じく「動的ルート」によりそれぞれの科目ごとのファイルに保存されるようにした。
- 科目ページ表示の際に、作成されたシラバスがあるか、既存のシラバスの URL があるか、なければ無いと表示するようにした。

効果

- シラバスの作成と適応が適切に行われるようになった。

やったこと 4

問題

- 上記の構造改革の過程で、実際のデータに潜む様々な「不整合」が原因で、多くのエラーが発生した。
- 例：
  ファイルごとの文字コードの混在 (UTF-8, CP932)
  CSV ファイル内の重複した列、意図しない空の列
  列名の表記揺れ（HPnumber , HPNumber, text , Text）

やったこと 4

解決策

- 複数の文字コードに対応する CSV 読み込み関数を実装。
- 列名の違いを吸収する「正規化」処理や、重複列を自動削除するロジックをデータ読み込み時に組み込んだ。

効果

- フォーマットが不統一なデータに対してもエラーを起こしにくい、安定したプログラムが完成した。

今後の予定

課題

- 未確認の不具合の発見
- 新規性（過去の先輩の就職先・成績データに基づく履修指標提示機能）の追加

新規性について

提案手法 A：高履修率・高成績科目分析

- 「目標の企業に行った先輩たちは、どの科目を多く履修し、良い成績を収めていたのか？」を直接的に集計・ランキング化する、直感的でシンプルなアプローチ。

提案手法 B：特徴重要度を用いたキー科目抽出

- 履修履歴を「学生の特徴」、就職先を「結果」と見立て、「どの科目の履修が、その就職結果を最も強く説明・予測できるか」を機械学習モデルで統計的に分析するアプローチ。

提案手法 A：高履修率・高成績科目分析

解析フロー

- 目標の指定
  ユーザーが目標とする企業や業種を選択する。
- 卒業生コホートの抽出
  就職先データから、指定されたキャリアパスに進んだ卒業生グループを特定する。
- 履修データの集計
  抽出された卒業生グループの履修履歴データを横断的に集計。各科目について「履修者数」と「優評価（◎）の割合」を計算する。
- スコアリングと推薦
  上記の集計結果から、独自のスコアを算出して科目をランキング化。ランキング上位の中から、ユーザーが未履修の科目を推薦する。

提案手法 A：高履修率・高成績科目分析

特徴

- 長所：
  ロジックが直感的で、結果の解釈が非常に容易である。
  実装が比較的簡単で、ベースラインとして有効。
- 短所：
  単純に履修人数が多いだけの必修科目などが、不当に高く評価される可能性がある。
  複数の科目の組み合わせといった、より深い関連性を捉えることが難しい。

提案手法 B：特徴重要度を用いたキー科目抽出

解析フロー

- データの前処理
  全学生の履修データを、学生を行、科目を列とする「特徴量行列」に変換する（履修済みなら 1, 未履修なら 0）。
- モデルの学習
  「目標の企業への就職」を目的変数（正解ラベル：就職なら 1, それ以外なら 0）とする。特徴量行列と目的変数を使って、機械学習の分類モデル（例：ロジスティック回帰、ランダムフォレスト）を学習させる。
- キー科目の抽出
  学習済みモデルから、各科目（特徴量）がどれだけ就職結果の予測に貢献したかを示す「特徴量重要度（Feature Importance）」を算出する。
- 推薦
  算出された重要度が高い順に科目をランキング化し、これを「キー科目」としてユーザーに推薦する。

提案手法 B：特徴重要度を用いたキー科目抽出

特徴

- 長所：
  単純な履修人数に左右されず、キャリアパスとの関連性が統計的に強い本質的な科目を抽出できる。
  マイナーだが特定の職種で極めて重要な科目、といった隠れた関係性の発見が期待できる。
  研究としての新規性・独創性を強くアピールできる。
- 短所：
  計算コストが比較的高く、結果の解釈にモデルに関する知識が必要となる。