

Technická dokumentace algoritmů pro identifikaci latentních struktur

Popis předpokladů, řetězení zpracování a příklad použití

Nežli přejdeme k popisu zpracování dat, je nezbytné popsat předpoklady vstupních dat. Na vstupu předpokládáme 5 datových souborů: vstupní testy, vstupní dotazník, semestrální data, dotazník na konci semestru a data ze zkouškového období. U datových souborů vstupních dat a dotazníků je jediný podmíněný sloupec, sloupec ID_STUDIA, který musí být přítomen ve všech datasetech a obsahuje jedinečný identifikátor jedince. Ostatní sloupce mohou mít libovolný název s jednou podmínkou danou identifikací multiselect sloupce. Ty mohou mít jakýkoliv název, ale zakončeny musí být celým číslem podle pořadí odpovědi. Ty sloupce, které takto identifikovány být nemají naopak ukončení celým číslem obsahovat nesmí. Z pohledu hodnot je důležité pro správnou identifikaci ordinálních hod dodržet formát hodnot jako „číslo – text“, např. 1 – Naprosto souhlasím, 2 – Souhlasím, 3 – Nesouhlasím, 4 – Naprosto nesouhlasím. Při dodržení těchto dvou podmínek jsou sloupce správně identifikovány pro další zpracování. U semestrálních a zkouškových dat je struktura více definovaná. U semestrálních dat kromě ID_STUDIA musí dataset obsahovat sloupce:

- PREDMKOD – kód předmětu
- PREDMNAZEV – název předmětu
- NAZEV_POLOZKY – název akce, např. 1. průběžný test, 2. elaborát, atd.
- BODY – udělená hodnota bodů, ale může sloužit i k jiným popisům hodnocení akce
- DATUM_AKCE – datum absolvování/odevzdání – vypočítává se týden

U zkouškových dat kromě ID_STUDIA musí dataset obsahovat sloupce:

- AK_ROK – počáteční rok akademického roku, např. pro 2023/2024 je to 2023
- SEMESTR – dvě možné hodnoty: ZS a LS
- PREDMKOD – kód předmětu
- NAZEV_PRED – název předmětu
- DATUM_PREDM – udělení zápočtu (je-li ukončením předmětu), datum absolvování zkoušky
- ZISKANO_KREDITU – počet získaných kreditů; při neúspěšném pokusu předpokládáno 0 kreditů

Rozdíl ve zpracování je u dat typu semestrální a zkouškové výsledky. V modelu, kdy jsou data takto striktně oddělena je semestrálních dat využita funkce `prepare_interim_data`, která převede vstupní semestrální data do formátu pro další zpracování. Jedná se o zavedení specifických příznaků (sloupců) kódovaných jako `TÝDEN_PREDMKOD_NAZEV_POLOZKY`, kdy hodnotou je počet získaných bodů. Každý jedinec je tak v datech reprezentován jedním řádkem (to je platné

pro všechny zpracovávané datasety). U zkouškových dat byl využit přístup extrakce příznaků přes modelování studijních trajektorií (funkce `prepare_trajectories`). Studijní trajektorie jsou graf tvořeny časově definovanými úrovněmi (v našem případě týden), který reprezentuje výsledky každého jedince v daném týdnu, ale sumarizačně, tedy odráží i stav týdne předešlého. Proto přístup předpokládá znalost datumu absolvování kredity hodnocenými událostí, které pak převedou na týden (funkce `input_weeks`). Tato reprezentace je použitelná pro zařazení všech zkouškových předmětů nebo pouze pro výběr (např. těch nejdůležitějších) předmětů. Následně jsou studijní trajektorie reprezentovány jako vektor údajů s kódováním `TÝDEN_kód_předmětu`, kde kód předmětu může být přímo ze vstupních dat nebo (defaultně) je vytvořen specifický bezvýznamový kód (funkce `course_node_coding`). Ve výsledku tak opět vzniká reprezentace, kde je každý jedinec daný jedním řádkem datasetu (funkce `make_atomic_dataset`). Takto jsou všechny reprezentace vstupních dat sladěny a následné zpracování je vždy stejné.

V případě, že jsou kredity uděleny na základě průběžného hodnocení (není striktní oddělení) je pak využity funkce `fill_credits`, `input_weeks` a `prepare_interim_data`. Trajektorie se v tomto případě nevyužívají. Ale je možné je napočítat a využívat, např. za účel vizualizace průchodů studiem. Jen je v tomto případě nutné počítat s vysokou variabilitou, protože kombinací různých akcí na předmět bude mnohem více než samotných předmětů (zkoušek). Tento fakt lze odstranit vynecháním položek a úpravou datasetu, to je však již mimo rámec připravených funkcí.

První částí je předzpracování dat. Všechny funkce pro tuto část jsou dostupné v části `learany.preprocessing.validation` a `learany.preprocessing.profile_columns`. Pro zpracování budeme postupovat pro každý dataset zvlášť. První krok který je nad každým datasetem proveden je validace dat pomocí funkce `validate`. Pokud jsou data v pořádku následuje odstranění prázdných řádek (jsou přítomny) funkcí `drop_empty_rows`, řídce vyplněných sloupců funkcí `drop_sparse_columns` a sloupců s volným textem funkcí `drop_free_text_columns`. Následuje zpracování multiselect sloupců. V prvním kroku jsou pomocí funkce `detect_ranked_multiselect_groups` nalezeni kandidáti takového typu sloupců. Ty jsou následně ohodnoceny funkcí `looks_like_multiselect` a pak finálně sestaveny do upraveného dataframe pomocí `multiselect_ranked_to_multihot`. Jako poslední krok zpracování multiselect sloupců je generování a export vah jednotlivých sloupců pomocí funkce `group_weights_from_prefixes`. Jelikož je zpracování multiselect sloupců v našem přístupu pro shlukování jediné, které rozšiřuje dataset, je nyní možné provést krok překódování názvů sloupců. Tento krok je zvolen s ohledem na komplikovanější názvy sloupců (např. s ohledem na diakritiku např. v dotazníkových datech). Tento krok je zajištěn funkcí `encode_columns`. Nyní je sestaven velký profil sloupců pomocí funkce `encode_columns`, který je pak využíván nadále v dalších krocích. Jako finální krok předzpracování jsou pak zakódovány hodnoty datasetu. Zde se jedná o sladění kódování binárních příznaků (ano/ne převedeno na 1/0), ordinální hodnoty jsou převedeny na čísla (1,2,3,...) a je zkontrolován jednotný směr významu.

Po předzpracování datasetu je jako první krok výběr příznaků pro shlukování. Všechny funkce pro tuto část jsou dostupné v části `learnany.clustering.clustering`. Tento proces je zastřešen funkcí `select_features_for_clustering`. V této funkci se využívá skóre jednotlivých příznaků vypočtené funkcí `structural_score`, tvořené variabilitou jednotlivých proměnných. Výsledek je rozdělen do hodnocení sloupců (proměnných) atributy YES, NO a MAYBE. V následném shlukování jsou defaultně použity proměnné s atributem YES. Následně jsou ve funkci `prepare_weights` doplněny váhy pro ostatní sloupce (neidentifikované jako multiselect). Po přípravě dat je vypočtena Gowerova vzdálenost, která je pro náš záměr vhodná. Tento výpočet je proveden funkcí `prepare_gower_distace`. Dataset je tímto procesem upraven do vhodného uspořádání pro shlukování. Následně je v rámci funkce `get_clusters` provedeno aglomerativní shlukování. To bylo zvoleno se záměrem hledat v datech ucelenější struktury. Počet shluků je určen na základě silhouet skóre v rozmezí 3 až 10 shluků.

Po provedení shlukování následuje proces extrakce strukturálních (popisných) pravidel nad jednotlivými shluky. Tyto funkce jsou umístěny v `learnany.clustering.cluster_rules`. Celý proces je zaštitěn dvěma funkcemi. První `rule_extart_data_prepare` připraví data pro extrakci pravidel a následně `pick_divers_rules` vybere pravidla, která jsou pak ke shluku přiřazena. Funkce `pick_divers_rules` obsahuje několik základních kroků. Nejprve jsou pomocí funkce `raw_rules_extractor` extrahovány všechna možná (pozorovaná) pravidla. Následně funkcí `rules_evaluation` proběhne evaluace pravidel několika mírami (např. síla podpory pravidla, jak moc pravidlo (subgroup) selektivně pokrývá jeden konkrétní shluk a ne ostatní, je shluk v rámci pravidla častější, než by bylo očekáváno z jeho velikosti). Tyto hodnocení jsou následně sloučena do skóre pravidla. Finální výběr pravidel může probíhat jedním ze dvou způsobů, funkcí `pick_diverse_rules_strict` nebo funkcí `pick_diverse_rules_fallback`. První z nich bere striktně nastavené limity pro zařazení pravidla. Funkce `pick_diverse_rules_fallback` umožňuje v případě, že nejsou splněny limity vybrat pravidla, která jsou limitlm nejbližší. V základním nastavení je využívána funkce `pick_diverse_rules_fallback`. Extrakcí strukturálních pravidel je zakončen proces hledání latentních struktur z dat.

Dalším krokem je připravit pohled na latentní struktury skrze informaci z „venčí“, např. třídu úspěchu/neúspěchu. Důležitou součástí tohoto procesu je vydefinování třídy. To může proběhnout zcela mimo, jen je potřeba třídu pak správně přiřadit, což znamená ve formátu dvojice [ID_STUDIA, CLASS] a přiřadit proměnné *dfic*. Pro studijní neúspěšnost jsou definovány dva možné přístupy, kdy může být třída zjištěna z dat. Oba jsou součástí `learnany.preprocessing.study_results`. Oba přístupy předpokládají znalost daných prahů pro neúspěšnost (v kódu je nastaveno 15 kreditů v zimním semestru a 30 kreditů v letním semestru). První přístup předpokládá odvození třídy z dat za zkouškové období, ve druhém je třída odvozena ze získávání kreditů v podstatě v průběhu. Důležité zde je, že pro tyto dva případy je určení třídy možné provést pomocí funkcí `create_class_by_data`. Pro účel hledání možné asociace struktur a

třídy jsou funkce v části `learn.preprocessing.clusters_and_rules`. Dvě základní funkce jsou `cluster_summary` a `cluster_rule_summary`. Funkce `cluster_summary` přiřazuje k definovaným shlukům (resp. jejich instancím) třídu a vypočítává out-of-fold (OOF) pravděpodobnosti pozitivní třídy pomocí stratifikované CV. Průměr OOF se pak přiřadí k popisu shluku. Z pohledu shluku jsou pak tři důležité parametry empirická pravděpodobnost (`p_emp`) pozitivní třídy, modelová pravděpodobnost pozitivní třídy (`p_model`) a poměr případů pozitivní třídy (`lift_vs_pop`). Čím podobnější jsou si `p_emp` a `p_model`, tím více je výsledek přesvědčivý a interpretace `lift_vs_pop` smysluplnější. Druhá funkce `cluster_rule_summary` je principiálně obdobná, ale uplatňuje pohled na pozitivní třídu přes kombinaci shluků a jeho odvozená strukturální pravidla.

Navazuje možnost vytvořit tzv. interpretační scénáře, které představují výběr těch nejsilnějších pravidel s rozšířeným slovním popisem. Tento proces je zašit funkčí `build_scenarios`, která využívá sestavuje kandidátní řešení a pak Jedná se o doplněk předchozích výstupů. Je nutné upozornit, že žádný scénářový případ nemusí být identifikován a tedy tento výstup může být prázdný.

Podobně je to s rozšířením shluku nebo pravidlem definované struktury o pravidla, která jsou vztažena ke třídě může být prázdný výstup. Tento proces je posledním krokem k popisu/ověření latentních struktur. Extrahovaná pravidla jsou zde již přímo vázána na třídu a tak rozšiřují celkový pohled na struktury. Extrakce kontextových pravidel je zastřešena funkcí `cherry_pick_class_rules`, v rámci které jsou nejprve vybrány vhodné shluky a pravidla (funkce `build_contexts`) a následně jsou pomocí funkce `mine_class_rules_in_context` extrahována doplňková pravidla vázaná na třídu.

Posledním typem zpracování je možnost identifikace přechodů jedinců mezi jednotlivými datasety. Samozřejmě je zde důležité správné řazení datových souborů. Zde je základem sestavení jedinců pro jednotlivé dvojice časových řezů (datasetu), kterou připravuje funkce `prepare_flow_panel`. Následně jsou spočteny přechody a podmíněné pravděpodobnosti (četnosti) pomocí funkce `compute_flows`. Pro očištění o řídké přechody a další artefakty je možné využít funkci `sanitize_flows`, která znormalizuje a očistí tabulku přechodů a znovu dopočítá pravděpodobnosti (tato funkce je však nepovinná). Po těchto krocích je pomocí funkce `make_sankey_figure` připraven Sankey diagram (přes knihovnu `plotly`) a výsledný přehled charakteristik uzlů a přechodů. Tyto mohou být přes modul `report%flows` a jeho funkci `md_to_pdf_report` být převedeny do pdf souboru.

Příklad použití:

Dotazníková data

```
# načtení souboru
file = pd.read_excel(r'cesta/soubor_dotazniku.xlsx')
file = file.rename(columns=lambda x: x.strip())
df_survey = file.copy()
# profilování sloupců a další přezpracování
df_survey_coded, profile_survey, weights_survey, col2code_survey, code2col_survey =
processing.preprocess_data(df_survey)
# hledání kandidátů na latentní struktury
X_survey, labels_survey, out_survey, picked_rules_survey =
processing.find_latent_structures(df_survey_coded, profile_survey, weights_survey,
code2col_survey)
# sestavení report pro shluky
cluster_report_survey = processing.get_cluster_report(profile_survey, X_survey,
labels_survey, code2col_survey, top_clusters = 5)
```

Semestrální data

```
# načtení souboru
file = pd.read_excel(r'cesta/semestralni_data.xlsx')
file = file.rename(columns=lambda x: x.strip())
df_semester = file.copy()
# předzpracování semestrálních dat
df_sem_prep = study_results.prepare_interim_data(df_semester)
# profilování sloupců a další přezpracování
df_sem_coded, profile_sem, weights_sem, col2code_sem, code2col_sem =
processing.preprocess_data(df_sem_prep)
# hledání kandidátů na latentní struktury
X_sem, labels_sem, out_sem, picked_rules_sem =
processing.find_latent_structures(df_sem_coded, profile_sem, weights_sem,
code2col_sem)
# sestavení report pro shluky
cluster_report_sem = processing.get_cluster_report(profile_sem, X_sem, labels_sem,
code2col_sem, top_clusters = 5)
```

Zkoušková data

```
# načtení souboru
file = pd.read_excel(r'cesta/zkouskova_data.xlsx')
file = file.rename(columns=lambda x: x.strip())
df_exam = file.copy()
# předzpracování zkouškových dat
dfu = study_results.filter_exam_data(df_exam, 2023)
```

```

# doplnění týdnů
dfr = study_results.input_weeks(df, weeks)
# překódování předmětů
df_student_course, selected_courses, code, coding =
study_results.course_node_coding(dfr, weeks, "ZS", selected_courses_wp)
# příprava trajektorií
traj, edges = study_results.prepare_trajectories(df_student_course, selected_courses,
code, coding, "ZS", 12, 21)
# převedení trajektorií do atomického tvaru
df_exam_rep = study_results.make_atomic_dataset(
    traj=traj,
    id_col="ID_STUDIA",
    week_col="TYDEN",
    coding=coding,
    week_from=14,
    week_to=20,
    binary=False
)
# profilování sloupců a další přezpracování
df_tr_coded, profile_tr, weights_tr, col2code_tr, code2col_tr =
processing.preprocess_data(df_exam_rep)
# hledání kandidátů na latentní struktury
X_tr, labels_tr, out_tr, picked_rules_tr =
processing.find_latent_structures(df_tr_coded, profile_tr, weights_tr, code2col_tr)
# sestavení report pro shluky
cluster_report_tr = processing.get_cluster_report(profile_tr, X_tr, labels_tr,
code2col_tr, 5)

```

Vytvoření dat pro pohled přes třídu

```

# přiřazení třídy jedincům přes zkoušková data
dfic = study_results.create_class_by_data(dfr, "ZS")

# Rozšíření datasetů
df_dec_survey = processing.prepare_extract_data(df_survey_1_coded, out_survey_1,
dfic, profile_survey_1)
df_dec_sem = processing.prepare_extract_data(df_sem_coded, out_sem, dfic,
profile_sem)
df_dec_tr = processing.prepare_extract_data(df_tr_coded, out_tr, dfic, profile_tr)

# Vztvoření pohledu pro shluky
df_cluster = clusters_and_rules.cluster_summary(
    df_dec[i],
    alpha=1.0 # Laplace smoothing - doporučeno
)

```

```

# Vytvoření pohledu pro shluky a pravidla
df_rules = clusters_and_rules.cluster_rule_summary(
    df_dec=df_dec[i],
    rules_df=rules_by_t[i],
    alpha=1.0,
    top_k_rules=7,          # max 5 pravidel na shluk
    min_support=10         # hranice pro flag_low_support
)

# Tvorba scénářů
scenarios_df, boundary_df = scenario.build_scenarios(
    df_cluster=df_cluster,
    df_rules=df_rules,     # výstup z cluster_rule_summary
    time_label=f"T{i}→final",
    cfg=cfg
)

# Přidání kontextových pravidel na třídu
contexts, rules = context_rules.cherry_pick_class_rules(df_dec[i], df_cluster,
df_rules, X_by_t[i].iloc[:, :-1], code2col[i],
context_rules.ContextCfg, depth=4, beam_width=30, top_k=20, n_bins=4)

Vytvoření popisu přechodů

# identifikace a popis přechodů
panel, time_cols = prepare_flows.prepare_flow_panel(out_by_t, MISSING_CLUSTER=6)
flows = prepare_flows.compute_flows(panel, time_cols)
flows = prepare_flows.sanitize_flows(flows)

# Vytvoření Sankey diagramu
fig, flows_used = prepare_flows.make_sankey_figure(flows, label_map=None,
min_count=5, title="Flows")
fig.show()

# Příprava reportu
atlas_md = flow_report.build_atlas_report_markdown(
    panel=panel,
    id_col="ID_STUDIA",
    time_cols=time_cols,
    flows_used=flows_used,
    rules_by_t=rules_by_t,
    params_fn=None      # může být None, nebo tvůj popis
)

```

Přehled funkcí

Název funkce	validate
Umístění	learn.preprocessing.validation
Účel funkce	Provede základní validaci a kontrolu kvality vstupního Pandas DataFrame. Funkce kontroluje základní „sanity“ vlastnosti dat (typ vstupu, prázdnot, duplicity názvů sloupců), ověřuje přítomnost požadovaných sloupců a heuristicky detekuje potenciální problémy, které mohou negativně ovlivnit další zpracování (např. clustering nebo modelování). Konkrétně funkce: - ověřuje, že vstup je neprázdný Pandas DataFrame, - kontroluje duplicitní názvy sloupců, - kontroluje přítomnost povinných sloupců, - detekuje řídce vyplněné sloupce podle podílu neprázdných hodnot, - detekuje pravděpodobné sloupce s volným textem, - detekuje skupiny ranked-multiselect sloupců podle názvů, - kontroluje minimální počet řádků a sloupců. Výsledkem je strukturovaný report obsahující chyby, varování a základní statistiky.
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame určený k validaci. required_columns: Iterovatelný seznam názvů sloupců, které musí být v DataFrame přítomny, nebo None. Pokud je zadán a některé sloupce chybí, je to reportováno jako chyba. min_rows (int, default=50): Minimální doporučený počet řádků. Při nižším počtu je přidáno varování. min_cols (int, default=1): Minimální požadovaný počet sloupců. Při nedodržení je přidána chyba. sparse_ratio (float, default=0.01): Minimální podíl neprázdných hodnot ve sloupci. Sloupce pod tímto prahem jsou považovány za řídce vyplněné a reportovány jako varování. free_text_min_nonnull (int, default=20): Minimální počet neprázdných hodnot, aby mohl být textový sloupec posuzován jako potenciální volný text.
Výstupní proměnné	Slovník (dict) s klíči: - "errors": seznam fatálních chyb, které mohou znemožnit další zpracování, - "warnings": seznam varování upozorňujících na možná rizika v datech,

	- "stats": základní statistiky datasetu (počet řádků, sloupců, počet řídkých sloupců, free-text sloupců a multiselect skupin).
Poznámky	Funkce sama data nemodifikuje, pouze analyzuje jejich strukturu a obsah. Sloupce s volným textem a ranked-multiselect skupiny jsou detekovány heuristicky a jejich identifikace nemusí být stoprocentně přesná.

Název funkce	drop_empty_rows
Umístění	learnany.preprocessing.profile_columns
Účel funkce	Odstraní řádky, které jsou prázdné ve všech sloupcích kromě prvního. Řádek je odstraněn, pokud jsou všechny hodnoty ve sloupcích od druhého po poslední `NaN`. První sloupec je při kontrole ignorován (typicky identifikátor nebo technický sloupec).
Vstupní proměnné	df (pandas.DataFrame): DataFrame vstupních dat
Výstupní proměnné	pandas.DataFrame: DataFrame bez řádků, které nemají žádná data ve sloupcích od druhého dál.

Název funkce	nonempty_ratio
Umístění	learnany.preprocessing.profile_columns
Účel funkce	Spočítá podíl neprázdných hodnot ve sloupci. Pro textové sloupce (object/string) považuje za neprázdné pouze hodnoty, které nejsou `NaN` a po oříznutí bílých znaků nejsou prázdným řetězcem. Pro netextové sloupce považuje za neprázdné všechny hodnoty různé od `NaN`.
Vstupní proměnné	s (pandas.Series): Vstupní sloupec.
Výstupní proměnné	float: Podíl neprázdných hodnot v intervalu <0.0, 1.0>.

Název funkce	drop_sparse_columns
Umístění	learnany.preprocessing.profile_columns
Účel funkce	Odstraní řídce vyplněné sloupce na základě podílu neprázdných hodnot. Pro každý sloupec spočítá zaplněnost pomocí `nonempty_ratio()` a odstraní sloupce, jejichž podíl neprázdných hodnot je menší než `min_ratio`.
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame. min_ratio (float, default=0.01): Minimální podíl neprázdných hodnot, který musí sloupec splňovat, aby nebyl odstraněn.
Výstupní proměnné	tuple:

	- pandas.DataFrame: DataFrame bez odstraněných sloupců. - list[str]: Názvy odstraněných sloupců. - pandas.Series: Zaplněnost všech sloupců (seřazeno vzestupně).
Poznámky	Definice „neprázdné“ hodnoty vychází z `nonempty_ratio()` (u textu se ignorují prázdné řetězce, u netextu pouze `NaN`).

Název funkce	looks_like_free_text
Umístění	learany.preprocessing.profle_columns
Účel funkce	Heuristicky určí, zda sloupec pravděpodobně obsahuje volný text. Sloupec je považován za „free text“, pokud: - je textového typu, - obsahuje alespoň `min_nonnull` neprázdných hodnot, - a zároveň má dostatečně vysokou unikátnost a průměrnou délku textu. Pokud počet unikátních hodnot přesáhne `max_unique_cap`, je sloupec automaticky označen jako volný text.
Vstupní proměnné	series (pandas.Series): Jeden vstupní sloupec ze vstupního DataFrame. min_nonnull (int, default=20): Minimální počet neprázdných hodnot pro vyhodnocení. unique_ratio_threshold (float, default=0.5): Minimální poměr unikátních hodnot k celkovému počtu neprázdných hodnot. min_avg_len (float, default=12.0): Minimální průměrná délka textu (v počtu znaků). max_unique_cap (int, default=5000): Pokud `nunique` překročí tuto hodnotu, sloupec je považován za volný text bez dalšího testování.
Výstupní proměnné	bool: True pokud sloupec vypadá jako volný text, jinak False.
Poznámky	Funkce ignoruje `NaN` a hodnoty, které jsou po `strip()` prázdné. Hodnotí pouze strukturální vlastnosti (unikátnost, délku), nikoli význam textu.
Název funkce	drop_free_text_columns
Umístění	learany.preprocessing.profle_columns
Účel funkce	Odstraní sloupce, které jsou heuristicky vyhodnoceny jako volný text. Sloupce jsou detekovány pomocí `looks_like_free_text()` a následně jsou odstraněny z DataFrame.
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame.
Výstupní proměnné	tuple: - pandas.DataFrame: DataFrame bez free-text sloupců. - list[str]: Názvy odstraněných sloupců.

Název funkce	<code>detect_ranked_multiselect_groups</code>
Umístění	<code>learn.preprocessing.profile_columns</code>
Účel funkce	Detekuje skupiny „ranked-multiselect“ sloupců na základě názvů sloupců. Hledá sloupce, které sdílejí společný základ názvu a liší se číselným suffixem (např. <code>`Hobby_1`</code>, <code>`Hobby_2`</code> nebo <code>`Hobby (1)`</code>, <code>`Hobby (2)`</code>). Takové sloupce seskupí pod společný klíč (base). Do výsledku ponechá pouze skupiny s alespoň <code>`min_cols`</code> sloupci a sloupce v rámci skupiny seřadí podle číselného suffixu.
Vstupní proměnné	<code>df</code> (<code>pandas.DataFrame</code>): Vstupní <code>DataFrame</code>. <code>min_cols</code> (<code>int</code>, <code>default=2</code>): Minimální počet sloupců ve skupině, aby byla považována za ranked-multiselect.
Výstupní proměnné	<code>dict[str, list[str]]</code> : Mapa <code>`base`</code> -> seřazený seznam sloupců patřících do skupiny.
Poznámky	Funkce pracuje čistě podle názvů sloupců (regex), nekontroluje obsah hodnot. Suffixy nemusí být souvislé (např. může existovat 1 a 3 bez 2).

Název funkce	<code>looks_like_multiselect</code>
Umístění	<code>learn.preprocessing.profile_columns</code>
Účel funkce	Heuristicky ověří, zda blok sloupců vypadá jako multiselect odpověď. Posouzení vychází z vlastností hodnot ve vybraných sloupcích: - blok nesmí být téměř prázdný (průměrný podíl neprázdných hodnot <code>>= `min_nonnull_ratio`</code>), - převážná část neprázdných hodnot by neměla být číselná (<code>`string_ratio` >= `min_string_ratio`</code>), - sloupce nesmí mít příliš vysokou kardinalitu (ochrana proti volnému textu).
Vstupní proměnné	<code>df</code> (<code>pandas.DataFrame</code>): <code>DataFrame</code> obsahující analyzované sloupce. <code>cols</code> (<code>list[str]</code>): Názvy sloupců tvořících jeden multiselect blok. <code>min_string_ratio</code> (<code>float</code>, <code>default=0.7</code>): Minimální podíl hodnot, které se nepodaří převést na číslo (tj. vypadají jako text/kategorie). <code>min_nonnull_ratio</code> (<code>float</code>, <code>default=0.05</code>): Minimální průměrný podíl neprázdných hodnot napříč blokem. <code>max_unique_per_col</code> (<code>int</code>, <code>default=200</code>): Maximální počet unikátních hodnot v jednotlivém sloupci; vyšší hodnoty typicky indikují volný text.
Výstupní proměnné	<code>bool</code> : <code>True</code> pokud blok sloupců odpovídá heuristice multiselectu, jinak <code>False</code> .
Poznámky	Funkce používá pokus o převod na čísla (<code>`pd.to_numeric`</code>) jako aproximaci toho, zda hodnoty vypadají jako numerické. Je to heuristika a může selhat na specifických kódováních.

Název funkce	<code>multiselect_ranked_to_multihot</code>
Umístění	<code>learany.preprocessing.profle_columns</code>
Účel funkce	Převede ranked-multiselect sloupce do multi-hot (binární) reprezentace. Z několika sloupců obsahujících volby (např. <code>`X_1`</code>, <code>`X_2`</code>, <code>`X_3`</code>) vytvoří binární indikátory pro každou unikátní volbu napříč sloupci. Pořadí (ranking) je v tomto převodu záměrně ignorováno. Nové sloupce jsou pojmenovány jako <code>`{prefix}__{volba}`</code>. Volitelně přidá také sloupec <code>`{prefix}__count`</code>, který obsahuje počet vybraných možností pro každý řádek.
Vstupní proměnné	<code>df</code> (<code>pandas.DataFrame</code>): Vstupní <code>DataFrame</code>. <code>cols</code> (<code>list[str]</code>): Názvy sloupců tvořících ranked-multiselect blok. <code>prefix</code> (<code>str</code>): Prefix pro názvy nových multi-hot sloupců. <code>drop_original</code> (<code>bool</code>, <code>default=True</code>): Pokud <code>True</code>, původní <code>`cols`</code> budou odstraněny. <code>add_count</code> (<code>bool</code>, <code>default=True</code>): Pokud <code>True</code>, přidá sloupec <code>`{prefix}__count`</code>.
Výstupní proměnné	<code>pandas.DataFrame</code> : <code>DataFrame</code> rozšířený o multi-hot sloupce (a případně <code>count</code>).
Poznámky	Prázdné řetězce jsou ignorovány stejně jako <code>`NaN`</code> . Pokud se stejná volba objeví ve více ranked sloupcích, výsledný indikátor zůstane 1.

Název funkce	<code>group_weights_from_prefixes</code>
Umístění	<code>learany.preprocessing.profle_columns</code>
Účel funkce	Vytvoří mapu vah pro multi-hot sloupce seskupené podle prefixů. Pro každý prefix <code>`p`</code> vyhledá sloupce ve tvaru <code>`p__*`</code> (kromě <code>`p__count`</code>) a rozdělí mezi ně celkovou váhu <code>`total_weight`</code> rovnoměrně. Pokud existuje sloupec <code>`p__count`</code>, přiřadí mu samostatnou váhu <code>`count_weight`</code>.
Vstupní proměnné	<code>df</code> (<code>pandas.DataFrame</code>): <code>DataFrame</code> obsahující multi-hot sloupce. <code>group_prefixes</code> (<code>iterable[str]</code>): Prefixy identifikující jednotlivé skupiny. <code>total_weight</code> (<code>float</code>, <code>default=1.0</code>): Celková váha rozdělená mezi dummy sloupce v rámci jedné skupiny. <code>count_weight</code> (<code>float</code>, <code>default=0.5</code>): Váha pro sloupec <code>`__count`</code> v rámci skupiny.
Výstupní proměnné	<code>dict[str, float]</code> : Slovník <code>`název_sloupce -> váha`</code> pro všechny nalezené sloupce.
Poznámky	Sloupce, které neexistují, jsou ignorovány. Funkce nemění <code>DataFrame</code> , pouze vrací váhovou mapu.

Název funkce	encode_columns
Umístění	learn.preprocessing.profile_columns
Účel funkce	Překóduje názvy sloupců na interní reprezentaci ve formátu Q<n>. Funkce vytvoří mapování původních názvů sloupců (kromě prvního sloupce) na kódy `Q1`, `Q2`, ... a přejmenuje DataFrame podle tohoto mapování. Zároveň vrací obě mapy (původní -> kód, kód -> původní) pro pozdější rekonverzi do původních názvů (např. při generování reportů).
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame. První sloupec (index 0) je ponechán beze změny; kódují se sloupce od druhého dál.
Výstupní proměnné	tuple: - pandas.DataFrame: DataFrame s přejmenovanými sloupci. - dict[str, str]: `col_to_code` mapování původní_název -> Q-kód. - dict[str, str]: `code_to_col` mapování Q-kód -> původní_název.
Poznámky	Interní kódy jsou generovány podle pořadí sloupců v DataFrame (pořadí je tedy významné a musí zůstat stabilní, pokud se mapy používají napříč kroky).

Název funkce	unique_nonnull
Umístění	learn.preprocessing.profile_columns
Účel funkce	Vrátí unikátní neprázdné hodnoty ze série jako novou pandas Series.
Vstupní proměnné	series (pandas.Series): Vstupní série.
Výstupní proměnné	pandas.Series: Série obsahující unikátní hodnoty z `series` bez `NaN`.
Poznámky	Výstupní pořadí odpovídá pořadí výskytu unikátních hodnot v datech.

Název funkce	is_yes_no_column
Umístění	learn.preprocessing.profile_columns
Účel funkce	Detekuje binární sloupec typu Ano/Ne. Sloupec je vyhodnocen jako Ano/Ne, pokud všechny neprázdné hodnoty (po oříznutí, převodu na lowercase a konverzi na string) patří do množiny {"ano", "ne"}.
Vstupní proměnné	series (pandas.Series): Vstupní sloupec.
Výstupní proměnné	bool: True pokud sloupec obsahuje pouze hodnoty Ano/Ne, jinak False.

Název funkce	is_binary_01_column
--------------	---------------------

Umístění	learnany.preprocessing.profle_columns
Účel funkce	Detekuje binární sloupec typu 0/1. Funkce se snaží převést neprázdné hodnoty na čísla. Pokud se podaří převést alespoň `min_numeric_ratio` neprázdných hodnot a unikátní numerické hodnoty jsou podmnožinou {0, 1}, sloupec je považován za binární 0/1.
Vstupní proměnné	series (pandas.Series): Vstupní sloupec. min_numeric_ratio (float, default=0.95): Minimální podíl neprázdných hodnot, které musí být převoditelné na číslo.
Výstupní proměnné	bool: True pokud sloupec odpovídá binárnímu 0/1, jinak False.

Název funkce	is_numeric_column
Umístění	learnany.preprocessing.profle_columns
Účel funkce	Detekuje numerický sloupec na základě převoditelnosti hodnot na číslo. Funkce se pokusí převést hodnoty na číslo (`pd.to_numeric`). Pokud je alespoň `min_nonnull_ratio` neprázdných hodnot převoditelných, sloupec je považován za numerický.
Vstupní proměnné	series (pandas.Series): Vstupní sloupec. min_nonnull_ratio (float, default=0.5): Minimální podíl převoditelných hodnot mezi neprázdnými hodnotami.
Výstupní proměnné	bool: True pokud sloupec vypadá jako numerický, jinak False.
Poznámky	Pokud je sloupec zcela prázdný (vše NaN), vrací False.

Název funkce	is_ordinal_dash_column
Umístění	learnany.preprocessing.profle_columns
Účel funkce	Detekuje ordinální sloupec kódovaný ve formátu „<číslo> - <popis>“. Funkce kontroluje, zda unikátní neprázdné hodnoty odpovídají regexu pro formát `N - text` (akceptuje také různé typy pomlček). Pokud podíl shod dosáhne alespoň `min_match_ratio`, sloupec je považován za ordinální.
Vstupní proměnné	series (pandas.Series): Vstupní sloupec. min_match_ratio (float, default=0.9): Minimální podíl unikátních hodnot, které musí odpovídat ordinálnímu vzoru.
Výstupní proměnné	bool: True pokud sloupec vypadá jako ordinální „N - text“, jinak False.

Název funkce	score_label
Umístění	learnany.preprocessing.profle_columns
Účel funkce	Ohodnotí text labelu jako pozitivní, negativní nebo nejasný.

	Používá jednoduchou slovníkovou heuristiku: pokud se v textu vyskytnou pozitivní klíčová slova (a současně ne negativní), vrátí +1; pokud se vyskytnou negativní klíčová slova (a současně ne pozitivní), vrátí -1; jinak 0.
Vstupní proměnné	text (str): Textový popis (label) ordinální hodnoty.
Výstupní proměnné	int: +1 pro „pozitivní“, -1 pro „negativní“, 0 pro „nejasné/smíšené“.
Poznámky	Heuristika je jazykově závislá (CZ) a nemusí fungovat pro jiné domény.

Název funkce	infer_ordinal_direction_from_labels
Umístění	learany.preprocessing.profle_columns
Účel funkce	Odhadne směr významu ordinální škály na základě textových labelů. Porovná text labelu u nejnižšího a nejvyššího kódu a pomocí `score_label()` rozhodne, zda „pozitivní“ význam je spíše u nízkých nebo u vysokých hodnot. Pokud není možné směr spolehlivě určit, vrátí "unknown".
Vstupní proměnné	code_to_label (dict[int, str]): Mapování `kód -> label` pro ordinální škálu.
Výstupní proměnné	tuple[str, float]: - direction (str): "low_is_positive", "high_is_positive" nebo "unknown". - confidence (float): Odhad spolehlivosti v intervalu <0.0, 1.0>.
Poznámky	Implementace používá pouze extrémy (nejnižší a nejvyšší kód). Pokud jsou labely smíšené/nejasné, vrátí "unknown".

Název funkce	extract_ordinal_mapping
Umístění	learany.preprocessing.profle_columns
Účel funkce	Extrahuje mapování kódů a labelů z ordinálních hodnot ve formátu „<číslo> - <text>“. Z unikátních neprázdných hodnot ve sloupci se pokusí parsovat formát `N - text` a vytvoří: - `code_to_label`: mapu `int kód -> text label` - `label_to_code`: mapu `text label -> int kód`
Vstupní proměnné	series (pandas.Series): Vstupní sloupec s ordinálními hodnotami.
Výstupní proměnné	tuple[dict[int, str], dict[str, int]]: - code_to_label: {kód: label, ...}

	- label_to_code: {label: kód, ...}
Poznámky	Pokud se stejné texty opakují pro různé kódy, v `label_to_code` vyhraje poslední zpracovaná hodnota.

Název funkce	profile_columns
Umístění	learn.preprocessing.profile_columns
Účel funkce	Vytvoří profil typů sloupců v DataFrame. Pro každý sloupec (kromě "ID_STUDIA") určí jeden z typů: - yes_no: binární Ano/Ne, - binary_01: binární 0/1, - numeric: numerický, - ordinal: ordinální ve formátu „N – text“, - categorical: ostatní (kategoriální / textové s nižší kardinalitou). Pro ordinální sloupce navíc vytvoří metadata: mapování kódů/labelů a odhad směru škály ("low_is_positive"/"high_is_positive"/"unknown").
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame.
Výstupní proměnné	dict: Slovník s klíči: - "yes_no", "binary_01", "numeric", "ordinal", "categorical": listy názvů sloupců, - "ordinal_meta": dict `sloupec -> metadata` pro ordinální sloupce.
Poznámky	Detekce typů je založena na heuristikách a pořadí kontrol je významné (Ano/Ne se testuje dříve než 0/1 atd.).

Název funkce	encode_values
Umístění	learn.preprocessing.profile_columns
Účel funkce	Normalizuje a překóduje hodnoty v DataFrame podle profilu typů sloupců. Transformace: - binární sloupce (Ano/Ne a 0/1) převede na numerické 1.0/0.0, - ordinální sloupce parsuje z formátu „N - text“ na numerické kódy a případně obrátí škálu tak, aby vyšší hodnota znamenala „pozitivnější“ (pokud to lze odhadnout), - kategoriální sloupce převede na string dtype.
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame, který bude modifikován (hodnoty ve vybraných sloupcích jsou přepsány). profile (dict): Profil sloupců vrácený funkcí `profile_columns()`.
Výstupní proměnné	pandas.DataFrame: DataFrame s překódovanými/normalizovanými hodnotami.

Poznámky	Funkce mění hodnoty ve `df` (přepis sloupců) a vrací stejný objekt/referenci. U ordinálních sloupců se očekává formát „N - text“; pokud parsování selže, vzniknou `NaN`. Směr ordinální škály je odhadován heuristicky z textů labelů a může být "unknown".
----------	---

Název funkce	prepare_interim_data
Umístění	learany.preprocessing.study_results
Účel funkce	Připraví pivotovanou mezitabulku s body po týdnech a položkách pro každé ID_STUDIA. Funkce vezme vstupní data v „long“ formátu, kde každý řádek typicky reprezentuje jednu položku hodnocení (BODY) pro konkrétní studium (ID_STUDIA), týden (TYDEN), předmětový kód (PREDMKOD) a název položky (NAZEV_POLOZKY). Data pivotuje do širokého tvaru tak, aby: - řádky odpovídaly unikátním hodnotám `ID_STUDIA`, - sloupce odpovídaly kombinacím (`TYDEN`, `PREDMKOD`, `NAZEV_POLOZKY`), - hodnoty v buňkách byly `BODY`. Pro vzniklé MultiIndex sloupce vytvoří jediné stringové názvy ve tvaru `TYDEN_PREDMKOD_NAZEV_POLOZKY` a nakonec vrátí DataFrame s `ID_STUDIA` jako běžným sloupcem (po `reset_index()`).
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame v dlouhém formátu. Očekávané sloupce: - "ID_STUDIA": identifikátor jednotky (řádek ve výsledné wide tabulce), - "TYDEN": týden (bude se snažit převést na int16), - "PREDMKOD": kód předmětu (převeden na string), - "NAZEV_POLOZKY": název položky (převeden na string), - "BODY": hodnota (bodové ohodnocení) použitá jako `values` v pivotu.
Výstupní proměnné	pandas.DataFrame: Wide DataFrame s jedním řádkem na `ID_STUDIA` a se sloupci odpovídajícími kombinacím týden/předmět/položka. Chybějící kombinace jsou doplněny nulou.
Poznámky	- `pivot_table(..., aggfunc="first")` předpokládá, že pro danou kombinaci klíčů je hodnota unikátní; pokud existuje více hodnot, vezme se první (podle interního pořadí). - `fill_value=0` vyplní chybějící hodnoty nulou, což je vhodné, pokud 0 znamená „bez bodů/bez záznamu“. Pokud 0 má jiný význam, je potřeba zvolit jiný přístup. - Název sloupce vzniká prostým spojením částí podtržítkem; pokud `PREDMKOD` nebo `NAZEV_POLOZKY` obsahují podtržítka, může to ztížit pozdější parsování názvů.

Název funkce	filter_exam_data
Umístění	learany.preprocessing.study_results
Účel funkce	Předfiltruje data o zkouškách/zápočtech pro konkrétní akademický rok a vyčistí neplatná hodnocení.
Vstupní proměnné	df_exam (pandas.DataFrame): Vstupní DataFrame se záznamy zkoušek/zápočtů. Očekávané sloupce minimálně: "AK_ROK", "HODNOCENI", "ID_STUDIA", "DATUM_PREDM". year: Hodnota akademického roku, který má být vybrán (typicky int nebo str podle reprezentace v datech).
Výstupní proměnné	pandas.DataFrame: Filtrovaný a seřazený DataFrame.

Název funkce	fill_credits
Umístění	learany.preprocessing.study_results
Účel funkce	Doplní získané kredity ke studijním předmětům na základě kumulativního součtu bodů. Funkce pro každého studenta a předmět seřadí záznamy chronologicky, vypočítá kumulativní součet bodů (sloupec 'CS') a při prvním dosažení nebo překročení zadaného limitu bodů zapíše do sloupce 'ZISKANO_KREDITU' počet kreditů daného předmětu. Ostatní hodnoty jsou doplněny nulou.
Vstupní proměnné	df (pandas.DataFrame): Vstupní DataFrame obsahující minimálně sloupce: 'ID_STUDIA', 'PREDMKOD', 'DATUM_AKCE', 'BODY'. course_credits (dict): Slovník mapující kódy předmětů ('PREDMKOD') na počet kreditů, které lze za předmět získat. limit (int): Hraniční hodnota kumulativního počtu bodů, při jejímž dosažení nebo překročení jsou studentovi přiděleny kredity.
Výstupní proměnné	pd.DataFrame: Kopie vstupního DataFrame rozšířená o sloupce: - 'CS' (kumulativní součet bodů) - 'ZISKANO_KREDITU' (přidělené kredity, 0 pokud limit nebyl dosažen)

Název funkce	input_weeks
Umístění	learany.preprocessing.study_results
Účel funkce	Doplní ke záznamům zkoušek/zápočtů číslo týdne pokusu ('TYDEN') na základě datumů a mapování akademických týdnů. Funkce: - provede robustní převod sloupce 'DATUM_PREDM' na datetime (neparsovatelné hodnoty → NaT), - určí ISO rok a ISO týden z data,

	- spočítá relativní týden pokusu vůči začátku akademického roku, - ošetří přesah zkouškového období do následujícího kalendářního roku, - použije fallback hodnoty týdne pro záznamy s neplatným nebo chybějícím datem, - odstraní všechna `ID_STUDIA`, u kterých se vyskytne neplatná hodnota týdne.
Vstupní proměnné	dfu (pandas.DataFrame): Vstupní DataFrame se záznamy zkoušek/zápočtů. čekávané sloupce minimálně: "DATUM_PREDM", "AK_ROK", "SEMESTR", "ID_STUDIA". weeks (dict): Slovník s mapováním akademických roků na týdny. Očekávané klíče: - "ak_year_week_start": {AK_ROK → první týden akademického roku}, - "week_year_end": {AK_ROK → poslední ISO týden akademického roku}, - "week_wp_exam_end": fallback týden pro ZS bez validního data, - "week_sp_end_2": fallback týden pro LS bez validního data.
Výstupní proměnné	pandas.DataFrame: Kopie vstupního DataFrame rozšířená o sloupec `TYDEN`, obsahující pouze validní záznamy.
Poznámky	- Hodnota `TYDEN` je nejprve inicializována na -1 a slouží jako indikátor neplatných nebo nekonzistentních záznamů. - Pokud se u některého `ID_STUDIA` objeví `TYDEN < 0`, jsou odstraněny všechny záznamy tohoto studia. - Výsledný sloupec `TYDEN` je typu `int`.

Název funkce	create_class_by_exam_data
Umístění	learany.preprocessing.study_results
Účel funkce	Vytvoří binární klasifikaci studentů na základě maximálního kumulativního zisku kreditů v daném semestru. Funkce: - vybere záznamy odpovídající zadanému semestru, - seřadí pokusy podle studenta a týdne pokusu, - spočítá kumulativní součet získaných kreditů pro každého studenta, - určí maximální dosažený kumulativní součet kreditů v semestru, - na základě prahové hodnoty přiřadí třídu studenta.

Vstupní proměnné	out (pandas.DataFrame): Vstupní DataFrame se záznamy zkoušek/zápočtů. Očekávané sloupce minimálně: "ID_STUDIA", "SEMESTR", "TYDEN", "ZISKANO_KREDITU". sem (str): Kód semestru, pro který má být klasifikace vytvořena. Očekávané hodnoty: "ZS" (zimní semestr) nebo "LS" (letní semestr).
Výstupní proměnné	pandas.DataFrame: DataFrame s jedním řádkem na `ID_STUDIA` a sloupcem `CLASS`, kde: - 0 značí dosažení nebo překročení kreditového prahu, - 1 značí nedosažení kreditového prahu.
Poznámky	- Prahová hodnota kreditů je nastavena na 15 pro "ZS" a 30 pro "LS". - Klasifikace vychází z maximální hodnoty kumulativního součtu kreditů dosažené v průběhu semestru, nikoli pouze z posledního pokusu. - Funkce implicitně předpokládá, že sloupec `ZISKANO_KREDITU` je číselného typu.

Název funkce	course_node_coding
Umístění	learany.preprocessing.study_results
Účel funkce	Připraví kódování předmětů (uzlů) pro tvorbu trajektorií student-týden a vrátí zjednodušenou tabulku se záznamy studenta o předmětech. Funkce: - vybere relevantní sloupce z `dfr` do `df_student_course`, - určí seznam předmětů (`selected_courses`) pro daný semestr (pokud není zadán), - zkontroluje duplicity v `selected_courses` a kapacitu kódovací abecedy, - přiřadí každému předmětu unikátní dvoupísmenný kód (AA, AB, ...), - vrátí mapování předmět → kód a také seznam použitých kódů.
Vstupní proměnné	dfr (pandas.DataFrame): Vstupní DataFrame se záznamy o pokusech/událostech. Očekávané sloupce minimálně: "ID_STUDIA", "NAZEV_PRED", "SEMESTR", "TYDEN", "HODNOCENI", "ZISKANO_KREDITU". weeks (dict): Slovník s informacemi o týdnech akademického roku. V této funkci se používá pouze pro odvození hranic týdne podle semestru. Očekávané klíče typicky: "week_wp_start", "week_wp_exam_end", "week_sp_start", "week_sp_exam_end_2". sem (str, optional): Kód semestru ("ZS" nebo "LS"). Default je "ZS". selected_courses (Iterable[str] None, optional): Explicitně zadáný seznam předmětů,

	které mají být zahrnuty. Pokud je None, použijí se všechny unikátní `NAZEV_PRED` v daném semestru.
Výstupní proměnné	tuple: - pandas.DataFrame: `df_student_course` se sloupci ["ID_STUDIA", "NAZEV_PRED", "SEMESTR", "TYDEN", "HODNOCENI", "ZISKANO_KREDITU"]. - list[str]: `selected_courses` (výsledný seznam předmětů zahrnutých do kódování). - dict[str, str]: `code` mapování {NAZEV_PRED → dvoupísmenný kód}. - list[str]: `coding` seznam použitých kódů ve stejném pořadí jako `selected_courses`.
Poznámky	- Kódy jsou generovány jako kartézský součin písmen A–Z (AA..ZZ), tj. max 676 předmětů. - Proměnné `week_start`/`week_end` jsou určeny podle `sem`, ale funkce je nevrací; slouží jen jako kontext pro navazující práci.

Název funkce	prepare_trajectories
Umístění	learany.preprocessing.study_results
Účel funkce	Vytvoří týdenní trajektorie studentů ve „wide“ tvaru a z nich odvodí přechody (hrany) mezi následnými stavy (uzly) v čase. Funkce: - odfiltruje data na zadaný semestr a vybrané předměty, - namapuje předměty na dvoupísmenné kódy (`CODE`) dle slovníku `code`, - agreguje data na úroveň (student, týden, kód) pomocí maxima kreditů v týdnu, - vytvoří kompletní mřížku všech týdnů pro každého studenta a doplní chybějící týdny, - zajistí kumulativní „stav“ po týdnech pomocí `ffill` (hodnota drží poslední známý stav), - doplní zbytek nulami a sestaví řetězcovou reprezentaci uzlu `REP`, - spočítá týdenní přírůstky kreditů a kumulativní součet (`CUMSUM`), - odvodí tabulku hran mezi uzly v týdnu t a t+1 včetně počtů studentů v uzlech.
Vstupní proměnné	df_student_course (pandas.DataFrame): Vstupní DataFrame se záznamy po studentech. Očekávané sloupce minimálně: "ID_STUDIA", "NAZEV_PRED", "SEMESTR", "TYDEN", "ZISKANO_KREDITU".

	<code>selected_courses</code> (Iterable[str]): Seznam předmětů (NAZEV_PRED), které mají být zahrnuty. <code>code</code> (dict[str, str]): Mapování {NAZEV_PRED → kód}, které se použije pro sloupec <code>`CODE`</code>. <code>coding</code> (list[str]): Seznam všech kódů, které mají existovat jako sloupce ve „wide“ tabulce. <code>semester</code>: Hodnota semestru, která se filtruje proti sloupci <code>`SEMESTR`</code> (např. "ZS"/"LS"). <code>week_from</code> (int): První týden (včetně), který má být vygenerován do mřížky týdnů. <code>week_to</code> (int): Horní mez týdne pro generování (<code>`np.arange(week_from, week_to)`</code>), tj. <code>`week_to`</code> je exkluzivní.
Výstupní proměnné	tuple: - <code>pandas.DataFrame</code>: <code>`wide`</code> na úrovni (ID_STUDIA, TYDEN) s: - sloupce kódů z <code>`coding`</code> (kumulativní stav, 0 pokud nic), - <code>`REP`</code> (řetězcová reprezentace stavu v týdnu), - <code>`ZISKANO_KREDITU`</code> (týdenní součet kreditů), - <code>`CUMSUM`</code> (kumulativní součet týdenních kreditů). - <code>pandas.DataFrame</code>: <code>`edges`</code> s přechody mezi uzly: - sloupce <code>`week`</code>, <code>`course1`</code>, <code>`course2`</code>, <code>`intersection_count`</code>, <code>`week_prev`</code>, <code>`count1`</code>, <code>`count2`</code>.
Poznámky	- Agregace na týden (student, týden, kód) používá maximum kreditů v týdnu jako „stav“. - Kumulativnost je zajištěna přes <code>`ffill`</code> po týdnech pro každého studenta; až poté se chybějící hodnoty doplňují nulami. - <code>`REP`</code> začíná prefixem týdne a dále obsahuje tokeny ve tvaru <code>"{CODE}{int_hodnota}"</code> oddělené tečkami (např. "5.AA0.AB3...").

Název funkce	<code>make_atomic_dataset</code>
Umístění	<code>learany.preprocessing.study_results</code>
Účel funkce	Převede kumulativní trajektorie na binární (0/1) nebo numerické atomické příznaky pro kombinace týden × kód. Funkce předpokládá vstupní tabulku, kde každý řádek odpovídá dvojici (uživatel, týden) a sloupce <code>`coding`</code> obsahují kumulativní stav (např. získané kredity) pro daný kód. Následně: - omezí data na týdny v intervalu [<code>`week_from`</code>, <code>`week_to`</code>], - zajistí existenci všech sloupců z <code>`coding`</code> (chybějící doplní nulami), - binarizuje hodnoty (<code>`> 0`</code> → 1, jinak 0), - přetaví data do „feature“ sloupců ve tvaru <code>"{week}.{code}"</code>, - zaručí kompletní sadu příznaků pro všechny týdny i kódy (chybějící doplní 0).

Vstupní proměnné	traj (pandas.DataFrame): Vstupní DataFrame s řádky (uživatel, týden) a se sloupci pro kódy v `coding`. Očekává se, že data jsou již kumulativní (ffill). id_col (str): Název sloupce s identifikátorem studenta (např. "ID_STUDIA"). week_col (str): Název sloupce s týdnem (např. "TYDEN"). coding (list[str]): Seznam kódových sloupců (např. ["AA", "AB", ...]). week_from (int): První týden (včetně), pro který se mají generovat příznaky. week_to (int): Poslední týden (včetně), pro který se mají generovat příznaky. keep_id_col (bool, optional): Pokud True, `id_col` zůstane jako běžný sloupec (po `reset_index()`); pokud False, vrátí se DataFrame s `id_col` jako indexem.
Výstupní proměnné	pandas.DataFrame: Binární feature-matice se sloupci "{week}.{code}" a hodnotami 0/1. Pokud `keep_id_col=True`, obsahuje i sloupec `id_col`.
Poznámky	- Binarizace pracuje s podmínkou `< 0`, takže jakákoli kladná hodnota znamená „aktivní“. - Výstup garantuje sloupce pro všechny kombinace týdnů a kódů v daném rozsahu, pokud se ve vstupních datech nevyskytují. - Sloupce jsou typu `int8` kvůli úspoře paměti.

Název funkce	binary_variability
Umístění	learany.clustering.clustering
Účel funkce	Ohodnotí variabilitu binárního příznaku na škále 0..1. Postup: - odstraní NaN, - pokusí se převést hodnoty na čísla (nepřevoditelné zahodí), - spočítá podíl jedniček `p` a vrátí $2 * \min(p, 1 - p)$. Interpretace: - 0.0 = žádná variabilita (vše 0 nebo vše 1, případně prázdné), - 1.0 = maximální variabilita ($p = 0.5$).
Vstupní proměnné	series (pandas.Series): Série s binárními hodnotami (0/1, případně stringy převoditelné na číslo).
Výstupní proměnné	float: Skóre variability v intervalu $<0.0, 1.0>$.
Poznámky	Hodnoty mimo {0, 1} nejsou explicitně zakázány; ovlivní `p` přes průměr. Pokud po čištění nezbude žádná hodnota, vrací 0.0.

Název funkce	ordinal_variability
Umístění	learany.clustering.clustering

Účel funkce	Ohodnotí variabilitu ordinálního příznaku na základě počtu unikátních hodnot. Skóre je definováno jako $(n_{\text{unique}} - 1) / \max(1, n_{\text{unique}})$, tj. roste s počtem různých úrovní a je 0 pro konstantní sloupec.
Vstupní proměnné	eries (pandas.Series): Série s binárními hodnotami (0/1, případně stringy převoditelné na číslo).
Výstupní proměnné	float: Skóre variability v intervalu $\langle 0.0, 1.0 \rangle$.
Poznámky	Hodnoty mimo $\{0, 1\}$ nejsou explicitně zakázány; ovlivní p přes průměr. Pokud po čištění nezbude žádná hodnota, vrací 0.0.

Název funkce	categorical_variability
Umístění	learnany.clustering.clustering
Účel funkce	Ohodnotí variabilitu kategoriálního příznaku pomocí dominance nejčastější kategorie. Skóre je definováno jako $1 - p_{\text{max}}$, kde p_{max} je relativní četnost nejčastější (nejdominantnější) kategorie. Hodnota 0.0 znamená, že všechny hodnoty patří do jedné kategorie, zatímco vyšší hodnoty značí větší rozmanitost kategorií.
Vstupní proměnné	series (pandas.Series): Série s kategoriálními hodnotami.
Výstupní proměnné	float: Skóre variability v intervalu $\langle 0, 1 \rangle$. Vrací 0.0, pokud je série prázdná nebo obsahuje pouze jednu kategorii.
Poznámky	Metrika je jednoduchá a snadno interpretovatelná, ale nezohledňuje počet kategorií ani jejich úplné rozložení (na rozdíl např. od entropie).

Název funkce	numeric_variability
Umístění	learnany.clustering.clustering
Účel funkce	Ohodnotí variabilitu numerického příznaku pomocí normalizovaného IQR. Skóre je definováno jako $IQR / (\max - \min)$, kde IQR je interkvartilové rozpětí. To dává robustní měřítko variability vůči outlierům, normalizované na rozsah dat.
Vstupní proměnné	series (pandas.Series): Série s numerickými hodnotami.
Výstupní proměnné	float: Skóre variability. Vrací 0.0, pokud je sloupec konstantní nebo má < 2 unikátní hodnoty.
Poznámky	Pokud je rozsah $\max - \min$ nulový, vrací 0.0 (ochrana proti dělení nulou).

Název funkce	structural_score
--------------	------------------

Umístění	learnany.clustering.clustering
Účel funkce	Spočítá strukturální skóre příznaku podle jeho typu a variability. Funkce nejprve spočítá typově odpovídající variabilitu (`*_variability`) a následně ji vynásobí váhou typu (`type_weight`). Skóre je použitelné jako jednoduché kritérium pro výběr sloupců do shlukování (vyšší = informativnější/variabilnější).
Vstupní proměnné	series (pandas.Series): Série s hodnotami příznaku. col_type (str): Typ příznaku. Očekávané hodnoty: "numeric", "ordinal", "binary", "categorical".
Výstupní proměnné	float: Strukturální skóre (0..~1), případně 0.0 pro neznámý typ.
Poznámky	Použité váhy jsou: numeric=1.0, ordinal=1.0, binary=0.7, categorical=0.6. Pro neznámé `col_type` je variabilita nastavena na 0.0.

Název funkce	select_features_for_clustering
Umístění	learnany.clustering.clustering
Účel funkce	Vyhodnotí a doporučí sloupce pro shlukování na základě strukturálního skóre. Pro každý sloupec v `column_types` spočítá `structural_score` a přiřadí rozhodnutí: - "YES" pokud score ≥ 0.25, - "NO" pokud score ≤ 0.05, - "MAYBE" jinak. Výsledek vrací jako DataFrame seřazený podle `structural_score` sestupně.
Vstupní proměnné	df (pandas.DataFrame): DataFrame s příznaky. column_types (dict[str, str]): Mapování `název_sloupce` -> typ` (např. "numeric", "binary", ...).
Výstupní proměnné	pandas.DataFrame: Tabulka se sloupci: - "column": název sloupce, - "type": typ sloupce, - "structural_score": skóre (zaokrouhlené na 3 desetinná místa), - "decision": doporučení ("YES"/"NO"/"MAYBE").
Poznámky	Sloupec "ID_STUDIA" je explicitně vynechán z hodnocení /rozhodnutí. Prah 0.25 a 0.05 jsou hard-coded; pokud mají být laditelné, je vhodné je parametrizovat.

Název funkce	gower_missing_aware_weighted
Umístění	learnany.clustering.clustering
Účel funkce	Spočítá váženou Gowerovu vzdálenost s podporou chybějících hodnot (NaN) a blokovým výpočtem.

	Funkce vytvoří matici párových vzdáleností D ($n \times n$) pro řádky DataFrame `X` kombinací: - numerické části: normalizovaná absolutní odchylka podle rozsahu ($\max - \min$) v každém sloupci, s ignorováním NaN a konstantních numerických sloupců, - kategoriální/binární části: 0 pokud se hodnoty rovnají, 1 pokud se liší, s ignorováním NaN. Pro každý pár (i, j) se vzdálenost počítá jako vážený průměr přes všechny platné (ne-NaN) příznaky; pokud pro dvojici neexistuje žádný platný příznak, vzdálenost je NaN. Výpočet probíhá po blocích řádků (`chunk_size`) kvůli paměti.
Vstupní proměnné	X (pandas.DataFrame): Vstupní data (řádky = objekty, sloupce = příznaky). numeric_cols (list): Seznam názvů numerických sloupců v `X`. cat_cols (list): Seznam názvů kategoriálních/binárních sloupců v `X`. weights (dict pandas.Series None): Váhy příznaků. Pokud None, všechny váhy = 1. Pokud je zadáno, reindexuje se na `X.columns` a chybějící se doplní 1.0. chunk_size (int, default=200): Velikost bloku řádků pro výpočet (trade-off paměť/čas).
Výstupní proměnné	numpy.ndarray: Symetrická matice vzdáleností tvaru (n, n) s nulovou diagonálou.
Poznámky	- Numerická normalizace používá rozsah `max-min` počítaný s `nanmin/nanmax`. Sloupce s nulovým rozsahem jsou vynechány (rozsah nastaven na NaN). - Kategoriální porovnání používá jednoduché `!=` nad objektovým polem. - Výsledek je po výpočtu symetrizován jako $(D + D.T) / 2$.

Název funkce	prepare_weights
Umístění	learnany.clustering.clustering
Účel funkce	Připraví datovou matici X, váhy příznaků a seznamy sloupců podle typu pro výpočet Gowerovy vzdálenosti. Funkce načte vybrané sloupce z `df` podle `df_clustering["column"]`, sestaví: - `X`: DataFrame pouze s vybranými featurami, - `weights`: slovník vah pro jednotlivé sloupce, - `cols`: slovník názvů sloupců rozdělených na numeric/ordinal/binary/categorical. Váhy: - výchozí váha je 1.0 pro všechny sloupce, - pokud jsou poskytnuty váhy pro multiselect (`weights_ms`), promítnou se do výsledku,

	- binární sloupce jsou shora omezeny na max 0.7, - kategoriální sloupce jsou shora omezeny na max 0.6.
Vstupní proměnné	df (pandas.DataFrame): Zdrojový DataFrame se všemi sloupci. df_clustering (pandas.DataFrame): Tabulka výběru pro clustering. Očekává sloupce: - "column": název featurey, - "type": typ featurey ("numeric", "ordinal", "binary", "categorical"). weights_ms (dict None): Volitelné váhy pro multiselect featurey (např. z prefixů).
Výstupní proměnné	tuple: - pandas.DataFrame: `X` s vybranými sloupci, - dict[str, float]: `weights` mapování sloupec -> váha, - dict[str, list[str]]: `cols` rozdělení sloupců podle typu.

Název funkce	prepare_gower_distance
Umístění	learany.clustering.clustering
Účel funkce	Wrapper pro výpočet vážené Gowerovy vzdálenosti nad připravenými typy sloupců. Sloučí numerické a ordinální sloupce do jedné množiny numerik (normalizovaná vzdálenost) a binární + kategoriální sloupce do množiny kategorií (0/1 shoda/neshoda).
Vstupní proměnné	X (pandas.DataFrame): Feature matice (řádky = instance, sloupce = featurey). cols (dict[str, list[str]]): Rozdělení sloupců podle typu, očekává klíče: "numeric", "ordinal", "binary", "categorical". weights (dict pandas.Series): Váhy jednotlivých sloupců.
Výstupní proměnné	numpy.ndarray: Matice vzdáleností D (n x n) pro aglomerativní shlukování.

Název funkce	get_clusters
Umístění	learany.clustering.clustering
Účel funkce	Provede aglomerativní shlukování nad předpočítanou maticí vzdáleností. Použije `AgglomerativeClustering` se vzdáleností zadanou jako "precomputed" a vazbou (linkage) "complete". Výsledkem je přiřazení shluků (labels) pro každou instanci.
Vstupní proměnné	D (numpy.ndarray): Čtvercová matice vzdáleností (n x n), typicky z Gowerovy metriky.
Výstupní proměnné	numpy.ndarray: Pole labelů délky n (cluster id pro každou instanci).
Poznámky	Při použití `distance_threshold` bývá v některých verzích sklearn nutné mít `n_clusters=None` (nebo naopak). Aktuální signatura

	předává obojí; je vhodné ověřit kompatibilitu s konkrétní verzí sklearn.
--	--

Název funkce	get_id_labels_frame
Umístění	learany.clustering.clustering
Účel funkce	Vytvoří DataFrame mapující ID_STUDIA na přiřazený shluk. Funkce vezme pole `labels` (v pořadí odpovídajícím řádkům `X`) a spojí jej se sloupcem "ID_STUDIA" z původního DataFrame `df` pro stejné indexy.
Vstupní proměnné	labels (array-like): Přiřazení shluků pro instance. X (pandas.DataFrame): Feature matice; používá se její index pro zarovnání s `df`. df (pandas.DataFrame): Původní DataFrame obsahující sloupec "ID_STUDIA".
Výstupní proměnné	pandas.DataFrame: DataFrame se sloupci ["ID_STUDIA", "cluster"].

Název funkce	rule_extart_data_prepare
Umístění	learany.clustering.cluster_rules
Účel funkce	Připraví datovou matici pro dolování subgroup pravidel nad shluky. Funkce: - pro všechny numerické/binary/ordinální sloupce přidá pomocné indikátory chybějících hodnot ve tvaru `{col}_is_nan` (0/1), - převede typy sloupců do formátů vhodných pro pravidlový extraktor: - numeric, binary -> float (přes to_numeric), - ordinal -> object (po numerickém převodu, aby se dal použít jako diskrétní), - categorical -> object, - přidá sloupec ``cluster`` s přiřazenými labely.
Vstupní proměnné	X (pandas.DataFrame): Feature matice (řádky = instance). labels (array-like): Labely shluků (délka = len(X)). cols (dict[str, list[str]]): Rozdělení sloupců podle typu, očekává klíče "numeric", "binary", "ordinal", "categorical".
Výstupní proměnné	pandas.DataFrame: Upravený DataFrame `X_prep` obsahující původní featury, pomocné ``*_is_nan`` indikátory a sloupec ``cluster``.
Poznámky	Funkce nastavuje globální pandas option `display.max_colwidth`, což ovlivní celé prostředí. Pokud to není žádoucí, je lepší to přesunout mimo funkci.

Název funkce	raw_rules_extractor
--------------	---------------------

Umístění	<code>learnany.clustering.cluster_rules</code>
Účel funkce	Extrahuje surová (neočištěná) diskriminační pravidla pro jednotlivé shluky pomocí subgroup discovery a beam search. Pro každý shluk ve sloupci <code>`cluster`</code> je spuštěna samostatná úloha subgroup discovery s binárním cílem „patří / nepatří do shluku“. Pro numerické příznaky jsou generovány intervalové selektory, pro kategoriální příznaky rovnostní selektory s omezenou kardinalitou.
Vstupní proměnné	<code>X</code> (<code>pandas.DataFrame</code>): Vstupní data obsahující příznaky a sloupec <code>`cluster`</code>. <code>numeric_cols</code> (<code>list[str]</code>, optional): Seznam numerických sloupců, pro které se vytvoří intervalové selektory. <code>cat_cols</code> (<code>list[str]</code>, optional): Seznam kategoriálních sloupců, pro které se vytvoří rovnostní selektory. <code>nbins</code> (<code>int</code>, default=5): Počet binů použitých při diskretizaci numerických proměnných. <code>beam_width</code> (<code>int</code>, default=30): Šířka beam search (počet kandidátů udržovaných v každé hloubce). <code>max_depth</code> (<code>int</code>, default=4): Maximální hloubka (počet selektorů) v jednom pravidle. <code>top_n</code> (<code>int</code>, default=30): Počet nejlepších pravidel vrácených pro každý shluk. <code>max_cat_values</code> (<code>int</code>, default=40): Maximální počet kategoriálních hodnot na jeden sloupec, pro které se generují selektory (omezení explozivního růstu search space).
Výstupní proměnné	<code>pandas.DataFrame</code>: Datový rámec s extrahovanými pravidly pro všechny shluky. Obsahuje metriky kvality pravidel a sloupec <code>`cluster_target`</code>, který určuje cílový shluk.
Poznámky	- Sloupec <code>`cluster`</code> je automaticky ignorován při generování selektorů. - Sloupce končící na <code>`_is_nan`</code> jsou také ignorovány. - Funkce neprovádí žádné filtrování, deduplikaci ani interpretaci pravidel; vrací pouze „raw“ výstup extraktoru. - Numerické sloupce jsou převáděny pomocí <code>`pd.to_numeric(errors="coerce")`</code>, kategoriální na typ <code>`category`</code>.

Název funkce	<code>eval_rule</code>
Umístění	<code>learnany.clustering.cluster_rules</code>
Účel funkce	Vyhodnotí jedno subgroup pravidlo pro cílový shluk <code>k</code> a spočítá metriky kvality. Pro zadaný subgroup (<code>sg</code>) spočítá: - coverage (kolik řádků pravidlo pokrývá),

	- confidence pro cílový shluk k (podíl k v pokrytí), - lift vůči base rate shluku, - dominance (confidence_k minus nejlepší confidence jiného shluku), - support_in_target (pokrytí v rámci cílového shluku), - cov_share (pokrytí vůči celé populaci), - agregované skóre kombinující čistotu, pokrytí a jedinečnost.
Vstupní proměnné	df (pandas.DataFrame): DataFrame obsahující sloupec "cluster" a featury. sg: Subgroup objekt (musí podporovat metodu `covers(df)` vracující boolean masku). k: Cílový shluk (hodnota v df["cluster"]). clusters (iterable): Seznam všech shluků pro výpočet dominance.
Výstupní proměnné	dict None: Slovník metrik (coverage, confidence_target, lift_target, ...), nebo None pokud je pokrytí nulové.
Poznámky	Skóre je kombinací více složek s pevnými vahami; pokud má být laditelné, je vhodné váhy/prahy parametrizovat.

Název funkce	rules_evaluation
Umístění	learany.clustering.cluster_rules
Účel funkce	Vyhodnotí sadu vytěžených pravidel napříč shluky a vrátí tabulku metrik. Pro každý řádek v `rules_raw` očekává: - "subgroup": objekt pravidla, - "cluster_target": cílový shluk, - případně "quality" (např. WRAcc) z dolovacího algoritmu. Následně pro každé pravidlo zavolá `eval_rule` a uloží výsledné metriky do jednoho DataFrame.
Vstupní proměnné	df (pandas.DataFrame): DataFrame s featurami a sloupцем "cluster". rules_raw (pandas.DataFrame): Kandidátní pravidla (např. z `raw_rules_extractor`). clusters (iterable): Seznam všech shluků.
Výstupní proměnné	pandas.DataFrame: DataFrame s metrikami pro každé pravidlo (včetně "rule_str").

Název funkce	pick_diverse_rules_strict
Umístění	learany.clustering.cluster_rules
Účel funkce	Vybere diverzní sadu pravidel pro shluk k za použití striktních filtrů a omezení podobnosti. Postup: - vyfiltruje pravidla pro cílový shluk `k`,

	- aplikuje minimální prahy na confidence, support a celkové pokrytí, - seřadí podle `score` sestupně, - iterativně vybírá pravidla tak, aby nebyla příliš podobná již vybraným ($Jaccard \leq max_jaccard$).
Vstupní proměnné	X (pandas.DataFrame): DataFrame, nad kterým lze vyhodnotit pokrytí pravidel (`covers`). rule_eval (pandas.DataFrame): Tabulka vyhodnocených pravidel (z `rules_evaluation`). k: Cílový shluk. n_rules (int, default=5): Kolik pravidel vybrat. min_conf (float, default=0.35): Minimální confidence pro cílový shluk. min_support (float, default=0.05): Minimální support v rámci cílového shluku. min_cov_share (float, default=0.005): Minimální pokrytí vůči celé populaci. max_jaccard (float, default=0.80): Maximální povolená Jaccardova podobnost vybraných pravidel.
Výstupní proměnné	pandas.DataFrame: Vybraná pravidla (podmnožina `rule_eval`) pro shluk k.

Název funkce	pick_diverse_rules_fallback
Umístění	learany.clustering.cluster_rules
Účel funkce	Vybere diverzní pravidla pro shluk k s fallbackem, pokud žádné neprojde přísnými prahy. Nejprve zkusí vybrat pravidla splňující (min_conf, min_support, min_cov_share). Pokud žádné neprojde, použije všechna pravidla pro daný shluk seřazená podle `score`. Při výběru stále hlídá diverzitu pomocí Jaccardovy podobnosti.
Vstupní proměnné	X (pandas.DataFrame): DataFrame pro vyhodnocení pokrytí pravidel. rule_eval (pandas.DataFrame): Vyhodnocená pravidla. k: Cílový shluk. n_rules (int, default=5): Počet pravidel k výběru. min_conf (float, default=0.45): Minimální confidence (pokud se použije striktní filtr). min_support (float, default=0.08): Minimální support v cílovém shluku. min_cov_share (float, default=0.01): Minimální pokrytí vůči populaci. max_jaccard (float, default=0.80): Maximální povolená podobnost vybraných pravidel.

Výstupní proměnné	pandas.DataFrame: Vybraná pravidla pro shluk k.
-------------------	---

Název funkce	replace_q_codes
Umístění	learany.clustering.cluster_rules
Účel funkce	Nahradí interní kódy typu Q<n> v textu pravidla jejich původními názvy. Funkce vyhledá tokeny odpovídající regexu `\\bQ\\d+\\b` a pro každý z nich provede substituci podle `mapping` (např. "Q15" -> "Název otázky"). Pokud kód v mapě chybí, ponechá původní token.
Vstupní proměnné	rule_str (str): Textová reprezentace pravidla. mapping (dict): Mapování `Q-kód -> původní název`
Výstupní proměnné	str: Text pravidla s nahrazenými kódy.

Název funkce	replace_q_codes_short
Umístění	learany.clustering.cluster_rules
Účel funkce	Nahradí Q-kódy v textu pravidla zkrácenou anotací „Qkód:krátký_název“. Pokud je kód v mapě, nahradí jej ve tvaru `Q<n>:<zkrácený_název` (přes `shorten()`). Pokud kód v mapě není, ponechá původní token.
Vstupní proměnné	rule_str (str): Textová reprezentace pravidla. mapping (dict): Mapování `Q-kód -> původní název`.
Výstupní proměnné	str: Text pravidla s nahrazenými kódy ve zkrácené podobě.

Název funkce	pick_divers_rules
Umístění	learany.clustering.cluster_rules
Účel funkce	Orchestruje dolování, vyhodnocení a výběr diverzních pravidel pro všechny shluky. Postup: 1) Zjistí seznam shluků z `X["cluster"]`, 2) vytěží kandidátní pravidla (`raw_rules_extractor`), 3) vyhodnotí pravidla metrikami (`rules_evaluation`), 4) vybere pro každý shluk `n_rules` diverzních pravidel: - striktně (`pick_diverse_rules_strict`) nebo s fallbackem (`pick_diverse_rules_fallback`), 5) volitelně vytvoří i lidsky čitelný text pravidel nahrazením Q-kódů názvy (plně nebo zkráceně).
Vstupní proměnné	X (pandas.DataFrame): DataFrame s featurami a sloupcem "cluster". numeric_cols (list[str] None): Numerické sloupce pro extraktor pravidel.

	<code>category_cols</code> (list[str] None): Kategoriální sloupce pro extraktor pravidel. <code>n_rules</code> (int, default=5): Počet pravidel na shluk. <code>min_conf</code> (float, default=0.45): Minimální confidence (u striktního/fallback filtru dle použité funkce). <code>min_support</code> (float, default=0.08): Minimální support v cílovém shluku. <code>min_cov_share</code> (float, default=0.01): Minimální pokrytí vůči celé populaci. <code>max_jaccard</code> (float, default=0.80): Maximální povolená podobnost vybraných pravidel. <code>fallback</code> (bool, default=False): Pokud True, použije fallback výběr. <code>original</code> (bool, default=False): Pokud True, vrátí i pojmenované/zpracované řetězce pravidel. <code>mapping</code> (dict None): Mapování Q-kódů na původní názvy (pro <code>`original=True`</code>). <code>short</code> (bool, default=False): Pokud True, použije zkrácenou náhradu (<code>`replace_q_codes_short`</code>).
Výstupní proměnné	<code>pandas.DataFrame</code> : Vybraná pravidla pro všechny shluky. Pokud <code>`original=True`</code> a <code>`mapping`</code> je dodáno, obsahuje navíc sloupec <code>"rule_str_named"</code> .

Název funkce	<code>_mask_interval</code>
Umístění	<code>learany.context.clusters_and_rules</code>
Účel funkce	Vytvoří booleovskou masku pro intervalovou podmínku s volitelně otevřenými/uzavřenými hranami. Interval je definován čísly <code>`a`</code> a <code>`b`</code> a závorkami: - <code>`lbr`</code> je <code>`['`</code> (včetně) nebo <code>`(`</code> (bez levé hranice), - <code>`rbr`</code> je <code>`]`</code> (včetně) nebo <code>`)`</code> (bez pravé hranice). Implementace podporuje i hodnotu <code>`['`</code> jako pravou závorku (kvůli historickému formátu v parseru), která je interpretována jako „otevřená“ pravá hranice (tj. <code>`< b`</code>).
Vstupní proměnné	<code>series</code> (<code>pandas.Series</code>): Numerická série. <code>a</code> (float): Levá hranice. <code>b</code> (float): Pravá hranice. <code>lbr</code> (str): Levá závorka <code>`['`</code> nebo <code>`(`</code>. <code>rbr</code> (str): Pravá závorka <code>`]`</code> nebo <code>`)`</code> (případně <code>`['`</code> dle parseru).
Výstupní proměnné	<code>pandas.Series</code> : Boolean maska <code>`a <= s <= b`</code> / varianty dle závorek.
Poznámky	

Název funkce	<code>_mask_cmp_numeric</code>
Umístění	<code>learany.context.clusters_and_rules</code>

Účel funkce	Vytvoří booleovskou masku pro porovnání numerické série s konstantou. Podporované operátory: ==, =, !=, <=, >=, <, >.
Vstupní proměnné	series (pandas.Series): Numerická série (ideálně již převedená přes `to_numeric`). op (str): Operátor porovnání. val (float): Porovnávaná hodnota.
Výstupní proměnné	pandas.Series: Boolean maska výsledku porovnání.
Poznámky	ValueError: Pokud je operátor nepodporovaný.

Název funkce	<code>_mask_cmp_str</code>
Umístění	<code>learany.context.clusters_and_rules</code>
Účel funkce	Vytvoří booleovskou masku pro porovnání stringové série s hodnotou. Pro stringy jsou povoleny pouze operátory rovnosti a nerovnosti (==, =, !=). Ostatní operátory jsou odmítnuty jako nesmyslné.
Vstupní proměnné	series (pandas.Series): Stringová série. op (str): Operátor porovnání (==, =, !=). val (str): Porovnávaný řetězec.
Výstupní proměnné	pandas.Series: Boolean maska výsledku porovnání.
Poznámky	ValueError: Pokud je operátor nepodporovaný pro string porovnání.

Název funkce	<code>eval_rule_str</code>
Umístění	<code>learany.context.clusters_and_rules</code>
Účel funkce	Vyhodnotí textové pravidlo (konjunkci podmínek) a vrátí masku řádků, které pravidlo splňují. Podporované typy podmínek (oddělené tokenem "AND"): 1) Interval: `feature: [a:b]` nebo `feature: (a:b)` a kombinace otevřených/uzavřených hran. Číslo `a`, `b` jsou parsována jako float, feature je vyhodnocena numericky. 2) Porovnání: `feature <op> value`, kde: - <op> je jeden z `==, =, !=, <=, >=, <, >`, - value je buď číslo (float), nebo quoted string ("..."/"..."). Numerické porovnání používá `_mask_cmp_numeric`, string porovnání `_mask_cmp_str`. 3) Null test: `feature.isnull()` nebo `feature.notnull()`. Výsledkem je maska vzniklá logickým AND přes všechny podmínky.

	Neparsovatelné hodnoty se převádí na NaN a jsou vyhodnoceny jako False.
Vstupní proměnné	df (pandas.DataFrame): DataFrame s featurami použitými v pravidle. rule_str (str): Textová reprezentace pravidla.
Výstupní proměnné	pandas.Series: Boolean maska (index zarovnaný s `df.index`) určující, které řádky splňují pravidlo.
Poznámky	- Podmínky se dělí přes literál "AND" (case-sensitive) a neřeší závorkování. - Pro string porovnání se používá dtype "string"; porovnání s NA vede na <NA>, proto se na konci každé podmínky používá `fillna(False)`.

Název funkce	prepare_pipe
Umístění	learany.context.clusters_and_rules
Účel funkce	Sestaví scikit-learn Pipeline pro predikci rizika se zpracováním chybějících hodnot. Pipeline obsahuje: - krok `AddMissingFlags`: přidá pro vybrané sloupce indikátory chybějících hodnot `{col}__missing` (0/1), - `ColumnTransformer` s imputací pro jednotlivé typy: - numerické a ordinální: konstantní imputace -1, - binární: konstantní imputace -1 + zahrnutí missing flagů, - kategoriální: imputace "__MISSING__" + OneHotEncoder, - klasifikátor `HistGradientBoostingClassifier`.
Vstupní proměnné	profile (dict): Profil sloupců (např. z `profile_columns()`), očekává klíče "numeric", "ordinal", "yes_no", "binary_01", "categorical".
Výstupní proměnné	sklearn.pipeline.Pipeline: Připravený pipeline objekt.

Název funkce	get_oof_proba
Umístění	learany.context.clusters_and_rules
Účel funkce	Spočítá out-of-fold (OOF) pravděpodobnosti pozitivní třídy pomocí stratifikované CV. Pro 10-fold stratifikovanou cross-validaci: - natrénuje `pipe` na trénovací části, - na trénu zvolí threshold maximalizující Youdenův index (aktuálně se však threshold nepoužije pro validaci; slouží jen jako pomocná informace), - uloží pravděpodobnosti pozitivní třídy na validačních indexech do `oof_proba`.
Vstupní proměnné	X (pandas.DataFrame): Feature matice.

	y (pandas.Series array-like): Binární cílová proměnná. pipe: scikit-learn pipeline s metodami <code>`fit`</code> a <code>`predict_proba`</code> .
Výstupní proměnné	numpy.ndarray: Pole délky <code>len(X)</code> s OOF pravděpodobnostmi; pro každý řádek je hodnota predikována modelem, který tento řádek neviděl při tréninku.

Název funkce	p_smooth
Umístění	learany.context.clusters_and_rules
Účel funkce	Laplaceovsky vyhlazený odhad pravděpodobnosti pro binární podíl. Vrací $(n1 + \alpha) / (n + 2 * \alpha)$, což odpovídá Beta(alpha, alpha) prioru a posterior mean pro Bernoulli data.
Vstupní proměnné	n1 (int): Počet pozitivních případů (y=1). n (int): Celkový počet případů. alpha (float, default=1.0): Síla vyhlazení (vyšší = silnější prior).
Výstupní proměnné	float: Vyhlazený odhad pravděpodobnosti; pokud <code>n == 0</code> , vrací <code>`np.nan`</code> .

Název funkce	cluster_summary
Umístění	learany.context.clusters_and_rules
Účel funkce	Vytvoří souhrn shluků s pohledem na cílovou třídu y a modelové OOF pravděpodobnosti. Pro každý shluk spočítá: - n a n1 (počet instancí a počet y=1), - p_emp (vyhlazený podíl y=1 přes <code>`p_smooth`</code>), - p_model (průměr OOF pravděpodobností ve shluku), - rozdíl p_emp - p_model, - lift vůči populaci (p_emp / p_pop), - směrodatnou odchylku OOF pravděpodobností ve shluku.
Vstupní proměnné	df_dec (pandas.DataFrame): DataFrame se sloupci minimálně: "cluster", "y", "oof_proba". alpha (float, default=1.0): Parametr Laplaceova vyhlazení pro <code>`p_smooth`</code> .
Výstupní proměnné	pandas.DataFrame: Souhrnná tabulka shluků seřazená podle velikosti shluku (n) sestupně.

Název funkce	cluster_rule_summary
Umístění	learany.context.clusters_and_rules
Účel funkce	Vytvoří souhrn shluků a jejich pravidel s pohledem na třídu y a OOF pravděpodobnosti. Pro každý shluk:

	- vybere top pravidla z `rules_df` (primárně podle `lift_target`, jinak podle `quality_wracc`), - pro každé pravidlo vyhodnotí masku nad `df_dec` (pomocí `eval_rule_str`) a spočítá metriky na průniku (cluster $\cap$ pravidlo): - p_emp_1 (vyhlazený podíl y=1), - p_model_1 (průměr OOF proba), - lift vůči shluku i vůči populaci, - confidence_target vůči třídě (raw n1/n), - lift_target vůči populaci (confidence / p_pop), - dominance vůči třídě v rámci shluku: $P(\text{rule} y=1, \text{cluster}) - P(\text{rule} y=0, \text{cluster})$, a doplní několik metrik z `rules_df` (quality_wracc, coverage, score).
Vstupní proměnné	df_dec (pandas.DataFrame): DataFrame se sloupci minimálně: "cluster", "y", "oof_proba" a featurami použitými v pravidlech. rules_df (pandas.DataFrame): Tabulka pravidel (např. z `pick_divers_rules`), očekává minimálně sloupce "cluster_target" a "rule_str" (případně "rule_str_named"). alpha (float, default=1.0): Parametr vyhlazení pro `p_smooth`. top_k_rules (int, default=5): Kolik pravidel vyhodnotit na shluk. min_support (int, default=10): Minimální support; pravidla pod tímto prahem jsou označena příznakem.
Výstupní proměnné	pandas.DataFrame: Souhrnná tabulka s řádky (cluster, pravidlo) a vypočtenými metrikami.

Název funkce	split_conditions
Umístění	learany.context. scenario
Účel funkce	Heuristicky určí, zda podmínka vypadá jako „negace“ nebo „absence“ signálu. Za „negativní“ považuje typicky: - testy na chybějící hodnotu (obsahuje "is_nan"), - nerovnost (!=), - rovnost s nulou (== 0), - případně porovnání typu "<= 0" (heuristika; může být doménově sporné).
Vstupní proměnné	cond (str): Jedna podmínka (bez AND), např. "Q5 == 0" nebo "Q7_is_nan == 1".
Výstupní proměnné	bool: True pokud podmínka vypadá jako „negativní“, jinak False.
Poznámky	Jde o čistě syntaktickou heuristiku – neinterpretuje význam featur.

Název funkce	is_negative_like_condition
Umístění	learany.context. scenario
Účel funkce	Spočítá podíl „negativně vypadajících“ podmínek v pravidle.

	Pravidlo se rozdělí přes `split_conditions` a pro každou podmínku se aplikuje `is_negative_like_condition`. Výsledek je `neg / n_conds`.
Vstupní proměnné	rule_eval (str): Text pravidla.
Výstupní proměnné	float: Podíl v intervalu <0.0, 1.0>. Pokud pravidlo nemá podmínky, vrátí 0.0.

Název funkce	rule_negativity_ratio
Umístění	learnany.context. scenario
Účel funkce	Vyhodnotí podíl negativně formulovaných podmínek v pravidle. Negativní podmínky jsou takové, které vylučují hodnoty (např. nerovnosti typu `!=`, `not in`, negace kategorií apod.). Skóre je definováno jako poměr počtu negativních podmínek k celkovému počtu podmínek v pravidle.
Vstupní proměnné	rule_eval (str): Textová reprezentace pravidla (např. ve formátu používaném subgroup discovery).
Výstupní proměnné	float: Podíl negativních podmínek v intervalu (0, 1). Vrací 0.0, pokud pravidlo neobsahuje žádné podmínky.
Poznámky	Vyšší hodnota skóre značí pravidla formulovaná spíše restriktivně („co neplatí“) než deskriptivně („co platí“), což může snižovat jejich interpretovatelnost.

Název funkce	rule_brief
Umístění	learnany.context. scenario
Účel funkce	Zkrátí text pravidla na maximálně `max_conds` podmínek a zbytek označí elipsou. Pokud je podmínek více než `max_conds`, vrátí prvních `max_conds` spojených " AND " a doplní suffix `AND ...(+k)`.
Vstupní proměnné	rule_named (str): Pravidlo (ideálně už „pojmenované“). max_conds (int, default=3): Max. počet podmínek, které se mají vypsát.
Výstupní proměnné	str: Zkrácený text pravidla.

Název funkce	default_why_text
Umístění	learnany.context. scenario
Účel funkce	Vytvoří jednoduché vysvětlení „proč“ z textu pravidla bez doménových znalostí. Jako vysvětlení použije první 1–2 podmínky z pravidla. Pokud podmínky nejsou, vrátí obecnou větu.
Vstupní proměnné	rule_named (str): Text pravidla (ideálně už pojmenované featury).
Výstupní proměnné	str: Krátký vysvětlující text.

Název funkce	scenario_score
Umístění	learany.context. scenario
Účel funkce	Spočítá skóre „scénáře pozitivní třídy (y=1)“ jako kombinaci rizikového zhoršení, podpory a stability. Základní princip: - spočítá zhoršení rizika pravidla vůči baseline shluku ($\delta_{emp} = p_{emp_rule} - p_{emp_cluster}$), - přidá support term (log nebo lineární) jako multiplikativní zesílení, - volitelně přimíchá modelovou informaci (p_{model_rule}) přes váhu <code>cfg.model_weight</code>, - penalizuje nesoulad empirie a modelu ($p_{emp_rule} - p_{model_rule}$) vůči toleranci <code>cfg.max_emp_model_gap</code> (hladkou penalizací).
Vstupní proměnné	<code>p_emp_rule</code> (float): Empirické riziko v rámci pravidla (typicky <code>p_emp_1</code>). <code>p_emp_cluster</code> (float): Empirické riziko celého shluku (<code>cluster_p_emp</code>). <code>support</code> (int): Počet instancí splňujících pravidlo (v rámci daného kontextu). <code>cfg</code> (ScenarioConfig): Konfigurace (váhy, prahy a volby výpočtu). <code>p_model_rule</code> (float None, optional): Průměrná modelová pravděpodobnost v pravidle (<code>p_model_1</code>).
Výstupní proměnné	float: Skóre scénáře (vyšší = zajímavější/„horší“ oproti shluku). Při nevalidních vstupech vrátí <code>np.nan</code> .

Název funkce	build_scenarios
Umístění	learany.context. scenario
Účel funkce	Sestaví „scénáře pozitivní třídy (y=1)“ a zároveň vymezující pravidla pro každý shluk. Vrací dvě tabulky: 1) <code>scenarios_df</code>: vybrané TOP scénáře (čas + shluk + pravidlo) splňující prahy na shluk i pravidlo, se skóre (scenario_score) a připraveným textovým „statementem“. 2) <code>boundary_df</code>: menší sada vymezujících/smíšených pravidel pro každý shluk (formální doplnění, i když pravidla nejsou nutně predikční). Hlavní kroky: - zvaliduje vstupy (povinné sloupce v <code>df_cluster</code> a <code>df_rules</code>),

	- sloučí statistiky shluku do pravidel (cluster_n, cluster_p_emp, cluster_lift_vs_pop), - vybere vymezující pravidla podle heuristiky „převaha negací“ (neg_ratio) a rankingu kvality, - identifikuje „rizikové shluky“ (min velikost + min p_emp + min lift vůči populaci), - z pravidel v rizikových shlucích vybere kandidáty scénářů podle prahů (support, p_emp_1, lift, stabilita empirie vs model), - spočítá `scenario_score` (kombinovaný delta emp/model + support term + penalizace gapu), - vybere top N pravidel na shluk (`cfg.max_rules_per_cluster`) a připraví reportovací sloupce.
Vstupní proměnné	df_cluster (pandas.DataFrame): Souhrn shluků. Povinné sloupce: "cluster", "n", "p_emp", "lift_vs_pop". df_rules (pandas.DataFrame): Souhrn pravidel (např. z `cluster_rule_summary`). Povinné sloupce: "cluster", "rule", "support", "p_emp_1", "p_model_1", "lift_vs_cluster", "flag_low_support", "rule_eval". (Další volitelné: "quality_wracc", "dominance", "score") time_label (str): Popisek času/okna (např. "W1", "W2", datum apod.) přidáný do výstupu. cfg (ScenarioConfig None): Konfigurace prahů a váh; pokud None, použije se default `ScenarioConfig`.
Výstupní proměnné	tuple[pandas.DataFrame, pandas.DataFrame]: - scenarios_df: reportovací tabulka scénářů (vybrané sloupce), - boundary_df: vymezující/smíšená pravidla pro shluky.

Název funkce	build_contexts
Umístění	learany.context.context_rules
Účel funkce	Sestaví přehled „kontextů“ (shluky a shluk+pravidlo) pro následné dolování třídních pravidel. Funkce vytváří dva typy kontextu: 1) cluster: samotný shluk Ck, pokud splňuje prahy na support, lift a (volitelně) stabilitu empirie vs model (` p_emp - p_model  <= cfg.max_emp_model_gap`). 2) rule: průnik (shluk $\cap$ strukturální pravidlo) podle tabulky `df_rules` – využije už předpočítané metriky (support, lift_vs_pop, emp_model_gap) a vybere jen nejrelevantnější. Výstupní tabulka je následně globálně omezena na `cfg.max_contexts_total` nejlepších kontextů (řazení podle lift_vs_pop a support).

Vstupní proměnné	df_dec (pandas.DataFrame): Datový rámec s instancemi (není přímo použit v těle funkce, ale je součástí signatury kvůli konzistenci pipeline). df_cluster (pandas.DataFrame): Souhrn shluků. Očekávané sloupce alespoň: "cluster", "n", "lift_vs_pop", "p_emp" (a volitelně "p_model"). df_rules (pandas.DataFrame): Souhrn pravidel v rámci shluků. Očekávané sloupce alespoň: "cluster", "support", "lift_vs_pop" a pro rule kontext také "rule" (zobrazení) a rule_eval" a "emp_model_gap". cfg (ContextCfg): Konfigurace prahů a limitů. Očekává atributy minimálně: `min_support`, `min_lift_vs_pop`, `max_emp_model_gap`, `max_contexts_total`.
Výstupní proměnné	pandas.DataFrame: Tabulka kontextů se sloupci typicky: - context_type ("cluster" "rule") - cluster (int) - context_id (str) - support (int) - lift_vs_pop (float) - gap (float) - rule_eval (str None) - rule_show (str None)
Poznámky	`gap` pro cluster kontext se počítá pouze pokud řádek obsahuje p_model; jinak je NaN.

Název funkce	context_mask
Umístění	learany.context.context_rules
Účel funkce	Vytvoří boolean masku řádků v `df_dec`, které spadají do zadaného kontextu. - Pro kontext typu "cluster" vrací masku `df_dec["cluster"] == k`. - Pro kontext typu "rule" vrací průnik masky shluku a masky vyhodnocené z `rule_eval` (pomocí `eval_rule_str`).
Vstupní proměnné	df_dec (pandas.DataFrame): DataFrame s instancemi a sloupcem "cluster" + featurami pro pravidla. ctx_row (pandas.Series): Jeden řádek z `build_contexts` (obsahuje "cluster", "context_type", a pro rule kontext "rule_eval").
Výstupní proměnné	pandas.Series: Boolean maska zarovnaná s `df_dec.index`.

Název funkce	make_ps_bins
Umístění	learany.context.context_rules
Účel funkce	Připraví featury do podoby vhodné pro pysubgroup: numeriky zabiní a vše převede na stringy bez inf hodnot.

	Pro každý sloupec ve `feature_cols`: - pokud je numerický dtype: - zkusí kvantilové binování přes `pd.qcut` (s `duplicates="drop"`), - při selhání zkusí `pd.cut`, - výsledný Interval převede na string a NaN vyplní "MISSING", - vytvoří nový sloupec `{col}__bin` a uloží jej do seznamu bin_cols, - pokud není numerický: - převede hodnoty na string a (zamýšleně) doplní missing "MISSING", - přidá sloupec do bin_cols beze změny názvu.
Vstupní proměnné	df (pandas.DataFrame): Vstupní data. feature_cols (list[str]): Sloupce, které se mají připravit pro dolování. n_bins (int, default=5): Maximální počet binů pro numerické sloupce.
Výstupní proměnné	tuple[pandas.DataFrame, list[str]]: - out: DataFrame s doplněnými bin sloupci a očištěnými inf hodnotami, - bin_cols: seznam sloupců, které jsou vhodné jako kategoriální pro pysubgroup.
Poznámky	Funkce nejprve nahradí $\pm\text{inf}$ za NaN (pysubgroup/numpy bývá na inf citlivý).

Název funkce	mine_class_rules_in_context
Umístění	learany.context.context_rules
Účel funkce	Vytěží pravidla pro cílovou třídu $y=1$ uvnitř zadaného kontextu (maska) pomocí pysubgroup. Postup: - vybere podmnožinu `df_dec[mask]` (kontext), - pokud je kontext menší než `min_support`, vrátí prázdný DataFrame, - připraví binované/kategoriální featury přes `make_ps_bins` (jen uvnitř kontextu), - sestaví search space selektorů (ignoruje vše mimo ps_cols + vybrané technické sloupce), - spustí beam search nad cílem `y == 1` s kvalitou WRAcc, - pro každé nalezené pravidlo spočítá support, p_emp_1, p_model_1, gap a lift vůči kontextu.
Vstupní proměnné	df_dec (pandas.DataFrame): DataFrame obsahující minimálně "y" (0/1), volitelně "oof_proba", a featury uvedené ve `feature_cols`. mask (pandas.Series): Boolean maska kontextu (typicky z `context_mask`). feature_cols (list[str]): Sloupce, které se mají použít jako kandidáti podmínek.

	depth (int, default=2): Maximální hloubka pravidla (počet selektorů). beam_width (int, default=50): Šířka beam search. top_k (int, default=5): Kolik nejlepších pravidel pysubgroup vrátí (result_set_size). min_support (int, default=20): Minimální support pravidla (uvnitř kontextu), jinak se zahodí. n_bins (int, default=5): Počet binů pro numerické featury.
Výstupní proměnné	pandas.DataFrame: Tabulka pravidel se sloupci typicky: "rule_class", "support", "p_emp_1", "p_model_1", "gap_emp_model", "lift_vs_context", seřazená dle ("lift_vs_context", "support") sestupně. Prázdná, pokud nic nenajde.

Název funkce	cherry_pick_class_rules
Umístění	learnany.context.context_rules
Účel funkce	Orchestruje výběr kontextů a dolování „třídních“ pravidel (y=1) v každém kontextu. Kroky: 1) vytvoří kontexty přes `build_contexts`, 2) pro každý kontext: - sestaví masku (`context_mask`), - nastaví adaptivní min_support (max(20, 10 % velikosti kontextu)), - vytěží pravidla přes `mine_class_rules_in_context`, - výsledky anotuje metadaty kontextu a přidá do společného seznamu.
Vstupní proměnné	df_dec (pandas.DataFrame): Data s instancemi (obsahuje "cluster", "y", volitelně "oof_proba"). df_cluster (pandas.DataFrame): Souhrn shluků (pro `build_contexts`). df_rules (pandas.DataFrame): Souhrn strukturálních pravidel (pro `build_contexts`). feature_cols (list[str]): Kandidátní featury pro dolování v kontextech. cfg_ctx (ContextCfg): Konfigurace prahů/limitů pro kontexty. depth (int, default=2): Hloubka pravidel. beam_width (int, default=50): Šířka beam search. top_k (int, default=10): Kolik pravidel dolovat v každém kontextu. n_bins (int, default=5): Počet binů pro numerické featury.
Výstupní proměnné	tuple[pandas.DataFrame, pandas.DataFrame]: - contexts: tabulka kontextů (z `build_contexts`),

	- all_rules: tabulka vytěžených pravidel napříč kontexty s metadaty kontextu.
--	---

Název funkce	<code>_pretty_num</code>
Umístění	<code>learany.context.context_rules</code>
Účel funkce	Zformátuje číselný string do „hezčí“ podoby pro textové výstupy. - převede na float, - hodnoty velmi blízké nule ($x < 1e-9$) zobrazí jako "0", - hodnoty velmi blízké celému číslu zobrazí jako int bez desetinné části, - jinak použije obecný formát <code>`g`</code>.
Vstupní proměnné	x (str): Hodnota jako string (nebo něco převoditelného na float).
Výstupní proměnné	str: Hezky zformátované číslo, nebo původní vstup, pokud převod selže.

Název funkce	<code>_parse_interval</code>
Umístění	<code>learany.context.context_rules</code>
Účel funkce	Naparsuje intervalový string ve formátu z pandas cut/qcut. Očekává formát typu: - "(a, b]" nebo "[a, b]" apod. (včetně případů s "inf" / "-inf"), - a vrátí čtveřici (left_bracket, a, b, right_bracket) jako stringy.
Vstupní proměnné	interval_str (str): Intervalový string, např. "(-0.001, 69.0]".
Výstupní proměnné	tuple[str, str, str, str] None: (lb, a, b, rb) pokud se podaří parsovat, jinak None.

Název funkce	<code>_feature_display_name</code>
Umístění	<code>learany.context.context_rules</code>
Účel funkce	Převede interní název featury na zobrazovaný název (včetně mapování Q-kódů). - odstraní suffix <code>"__bin"</code>, - pokud vzniklý název přesně odpovídá klíči v <code>`code2col`</code>, nahradí jej původním názvem, jinak vrátí nezměněný.
Vstupní proměnné	raw_feat (str): Název featury (např. "Q6__bin"). code2col (dict[str, str]): Mapování interního kódu na původní text.
Výstupní proměnné	str: Zobrazovaný název featury.

Název funkce	<code>condition_to_text</code>
Umístění	<code>learany.context.context_rules</code>
Účel funkce	Převede jednu podmínku (typicky z pysubgroup) do stručného lidského textu.

	Podporované vzory: - <code>`feat == 'value'` / `feat != 'value'`</code> (stringové hodnoty v apostrofech), přičemž pokud <code>`value`</code> vypadá jako interval <code>"(a, b]"</code> nebo <code>"[a, b]"</code>, převede to na větu o intervalu. - <code>`feat == number` / `feat != number`</code> (čísla). Zobrazení názvů feature: - odstraňuje <code>"__bin"</code>, - mapuje Q-kódy přes <code>`code2col`</code>.
Vstupní proměnné	<code>cond (str)</code>: Jedna podmínka jako string. <code>code2col (dict[str, str])</code>: Mapování interních názvů na původní text.
Výstupní proměnné	<code>str</code> : Podmínka přepsaná do čitelné věty. Pokud se vzor nerozpozná, vrátí původní <code>`cond`</code> .
Poznámky	Otevřenost/zavřenost hranic intervalu je ve větě zjednodušená (ignoruje se).

Název funkce	<code>rule_to_sentence</code>
Umístění	<code>learany.context.context_rules</code>
Účel funkce	Převede celé pravidlo ve tvaru 'A AND B AND C' na jednu lidsky čitelnou větu. Postup: - rozdělí pravidlo podle "AND", - každou část převede přes <code>`condition_to_text`</code>, - spojí části zvoleným spojovníkem (<code>`joiner`</code>), - pokud je podmínek více než <code>`max_terms`</code>, výstup zkrátí a doplní informaci o zbytku.
Vstupní proměnné	<code>rule_class (str)</code>: Text pravidla (např. z <code>pysubgroup</code>), typicky <code>"Q1__bin == '...' AND Q2 != 0"</code>. <code>code2col (dict[str, str])</code>: Mapování interních názvů featur na původní názvy. <code>intro (str, default="Platí, že")</code>: Úvod věty. <code>joiner (str, default=" a ")</code>: Spojka mezi podmínkami ve větě. <code>max_terms (int, default=8)</code>: Maximální počet podmínek vypsanych ve větě.
Výstupní proměnné	<code>str</code> : Jedna věta popisující pravidlo.

Název funkce	<code>prepare_flow_panel</code>
Umístění	<code>learany.context.prepare_flows</code>
Účel funkce	Sestaví „panel“ (wide tabulku) s přiřazením shluků pro stejné ID_STUDIA v několika časových řezech. Očekává <code>`out_by_t`</code> jako sekvenci (typicky délky 5) DataFrameů, kde každý obsahuje sloupce <code>["ID_STUDIA", "cluster"]</code> pro daný

	časový bod t0..tn. Data se postupně slučují přes outer join na "ID_STUDIA", čímž se zachovají i jedinci chybějící v některém čase. Chybějící shluky (NaN po merge) se doplní konstantou `MISSING_CLUSTER` a převedou na int.
Vstupní proměnné	out_by_t (Sequence[pandas.DataFrame]): Seznam/tabulka výstupů po časech; položky musí mít minimálně sloupce "ID_STUDIA" a "cluster". MISSING_CLUSTER (int, default=10): Kód shluku použitý pro chybějící časové body.
Výstupní proměnné	tuple[pandas.DataFrame, list[str]]: - panel: DataFrame se sloupci ["ID_STUDIA", "cluster_t0", ..., "cluster_tn"], - time_cols: seznam názvů časových sloupců ve správném pořadí.

Název funkce	compute_flows
Umístění	learany.context.prepare_flows
Účel funkce	Spočítá přechody mezi shluky mezi sousedními časovými body a odvodí podmíněné pravděpodobnosti. Pro každé t vytvoří tabulku počtů přechodů: count = # {ID: cluster_t == c_from a cluster_{t+1} == c_to} a doplní: - p_forward = P(c_to c_from, t) (normalizace v rámci (t, c_from)), - p_backward = P(c_from c_to, t) (normalizace v rámci (t, c_to)).
Vstupní proměnné	panel (pandas.DataFrame): Wide tabulka s časovými sloupci shluků (např. z `prepare_flow_panel`). time_cols (list[str]): Seznam názvů sloupců v pořadí času (např. ["cluster_t0", ..., "cluster_t4"]).
Výstupní proměnné	pandas.DataFrame: Tabulka přechodů se sloupci: - t (int): index přechodu (0 = t0->t1, 1 = t1->t2, ...), - c_from (int), c_to (int), - count (int), - p_forward (float), p_backward (float).

Název funkce	sanitize_flows
Umístění	learany.context.prepare_flows
Účel funkce	Znormalizuje a očistí tabulku přechodů a znovu dopočítá pravděpodobnosti. Funkce: - ověří přítomnost povinných sloupců ("t", "c_from", "c_to", "count"),

	- odstraní řádky s NaN v povinných sloupcích, - převede typy na numerické a následně na int (přes Int64 s dropna), - recompute: - p_forward = count / sum(count) v rámci (t, c_from), - p_backward = count / sum(count) v rámci (t, c_to).
Vstupní proměnné	flows (pandas.DataFrame): Vstupní tabulka přechodů (např. z `compute_flows`).
Výstupní proměnné	pandas.DataFrame: Očištěná kopie `flows` s garantovanými int typy pro t/c_from/c_to/count a s (p_forward, p_backward) znovu vypočtenými.
Poznámky	Pokud po čištění nezbudou žádné řádky, vrací prázdný DataFrame.

Název funkce	node_label_safe
Umístění	learany.context.prepare_flows
Účel funkce	Vrátí bezpečný popisec uzlu pro Sankey diagram (čas t + shluk c) s volitelným mapováním. Pokud je `label_map` dict a obsahuje klíč (t, c), použije jeho hodnotu jako label. Jinak vrátí výchozí tvar "t{t}:c{c}".
Vstupní proměnné	t (int): Časový index uzlu. c (int): ID shluku v čase t. label_map (dict[tuple[int,int], str] None): Volitelné mapování (t, c) -> label.
Výstupní proměnné	str: Textový label uzlu.

Název funkce	build_sankey_inputs
Umístění	learany.context.prepare_flows
Účel funkce	Připraví vstupy (nodes/links) pro Sankey diagram z tabulky přechodů. Kroky: - nejprve zavolá `sanitize_flows` (garance typů + dopočet pravděpodobností), - odfiltruje hrany s `count < min_count`, - sestaví množinu uzlů jako dvojice (t, cluster) pro zdroj i cíl (t+1), - vytvoří mapování uzel -> index (node_id) a list labelů (přes `node_label_safe`), - vytvoří listy `sources`, `targets`, `values` pro linky.
Vstupní proměnné	flows (pandas.DataFrame): Tabulka přechodů se sloupci minimálně t, c_from, c_to, count. label_map (dict None): Volitelné mapování (t, c) -> label pro uzly. min_count (int, default=1): Minimální počet přechodů, aby se hrana zobrazila.

Výstupní proměnné	tuple: - node_labels (list[str]): Labely uzlů v pořadí indexů, - sources (list[int]): Indexy zdrojových uzlů pro každou hranu, - targets (list[int]): Indexy cílových uzlů pro každou hranu, - values (list[int]): Hodnoty (počty) pro každou hranu, - f (pandas.DataFrame): Použitá (očistěná a filtrovaná) tabulka flows.
Poznámky	Pokud po čištění/filtrování nezbudou žádné hrany, funkce vrátí prázdné listy (nikdy None), aby volající kód nespádl.

Název funkce	make_sankey_figure
Umístění	learany.context.prepare_flows
Účel funkce	Vytvoří Plotly Sankey diagram pro přechody mezi shluky a vrátí i použitá data. Použije `build_sankey_inputs` pro konstrukci uzlů a hran. Pokud po filtrování nezbudou žádné hrany, vrátí prázdnou figuru s anotací „Žádné přechody...“.
Vstupní proměnné	flows (pandas.DataFrame): Tabulka přechodů (t, c_from, c_to, count, ...). label_map (dict None): Volitelné mapování (t, c) -> label pro uzly. min_count (int, default=1): Minimální count pro zahrnutí hrany do diagramu. title (str None): Volitelný titulek grafu.
Výstupní proměnné	tuple: - fig (plotly.graph_objects.Figure): Sankey figura (nebo prázdná s anotací), - f_used (pandas.DataFrame): Očištěná a filtrovaná flows tabulka použitá pro vykreslení.
Poznámky	Funkce předpokládá import `plotly.graph_objects as go`. Pavděpodobnosti (p_forward/p_backward) se do grafu přímo nepromítají, ale jsou dostupné ve `f_used` pro případné tooltipy / další analýzu.

Název funkce	prepare_report_md
Umístění	learany.reporting.cluster_report
Účel funkce	Vypočte profil shluků a vrátí jej jako markdown text (pro report). Funkce vezme matici příznaků `X` a přiřazení shluků `labels` a pro každý shluk spočítá jednoduché „odlišující“ charakteristiky vůči zbytku populace, odděleně pro typy proměnných: - Velikost shluku: `n` a podíl `share` v datasetu. - Numerické proměnné: rozdíl mediánu shluku vůči mediánu zbytku (`delta_median`).

	- Binární proměnné: rozdíl průměru (podílu 1) shluku vůči zbytku (<code>`delta`</code>). Před výpočtem se navíc pokusí převést hodnoty "Ano"/"Ne" na 1/0. - Ordinální proměnné: porovnání distribucí shluk vs. zbytek: - L1 vzdálenost mezi distribučními vektory (0..2), - KS-like metrika jako max rozdíl kumulativních distribucí (0..1), - mediány shluk / zbytek, - podíl „high“ (poslední 1–2 úrovně) a jeho rozdíl (<code>`delta_p_high`</code>). Do textu se zapisují jen ordinální proměnné, které překročí prahy (L1, KS, <code>delta_p_high</code>). - Kategoriální proměnné: top kategorie ve shluku a její rozdíl podílu vůči zbytku. Výstup se skládá do markdownu ve struktuře: - "# Profil shluků" - pro každý shluk "### Shluk k (n=..., ...%)" - podsekcce "#### Binární/Numerické/Ordinální/Kategoriální proměnné" s top <code>`count`</code> položkami.
Vstupní proměnné	X (pandas.DataFrame): Feature matice (řádky = instance). Musí být indexovatelná pomocí boolean masky. labels (array-like): Přiřazení shluků pro řádky X (délka = len(X)). cols (dict[str, list[str]]): Rozdělení sloupců podle typu; očekává klíče "numeric", "binary", "ordinal", "categorical" (klíče mohou chybět). count (int): Kolik nejvýraznějších proměnných na typ vypsát (top-N). code_to_col (dict): Mapování interních názvů featur (např. Q-kódy) na původní názvy pro report.
Výstupní proměnné	str: Markdown text popisující profily shluků.
Poznámky	- Funkce modifikuje `X` in-place pro binární sloupce (nahrazuje "Ano"/"Ne" a převádí na čísla). - Ordinální úrovně se berou jako sjednocení hodnot ve shluku a mimo shluk a poté se třídí. - <code>`np.unique(labels)`</code> určuje pořadí shluků ve výstupu; shluky se v textu nevypisují podle velikosti, ale podle hodnoty labelu.

Název funkce	<code>df_to_wrapped_table</code>
Umístění	<code>learany.reporting.rules_report</code>
Účel funkce	Převéde pandas DataFrame na ReportLab <code>`Table`</code> s „zalomenými“ texty a jednotným stylem. Funkce: - vytvoří dvě <code>`ParagraphStyle`</code> (hlavička/tělo) s fonty "CZ-Regular"/"CZ-Bold",

	- zkopíruje DataFrame a zaokrouhlí všechny numerické sloupce na <code>`round_digits`</code>, - hlavičku tabulky vloží jako <code>`Paragraph`</code> (kvůli zalamování a centru), - buňky v těle: - pokud je hodnota string, převede ji na <code>`Paragraph`</code> (zalamování), - jinak nechá hodnotu jako číslo (ReportLab ji vykreslí bez Paragraph), - nastaví <code>`TableStyle`</code> (barva hlavičky, fonty, zarovnání, mřížka, padding), - vrátí <code>`reportlab.platypus.Table`</code>, kterou lze vložit do <code>`story`</code>.
Vstupní proměnné	<code>df</code> (pandas.DataFrame): Vstupní tabulka k vykreslení. <code>styles</code>: ReportLab stylesheet (např. z <code>`getSampleStyleSheet()`</code>), používá se jako parent stylů. <code>col_widths</code> (int list, default=75): Šířky sloupců předané do <code>`Table(colWidths=...)`</code>. Může být jedno číslo (aplikuje se na všechny sloupce) nebo seznam šířek. <code>round_digits</code> (int, default=2): Počet desetinných míst pro zaokrouhlení numerických sloupců.
Výstupní proměnné	<code>reportlab.platypus.Table</code> : Naformátovaná tabulka připravená k vložení do PDF.
Poznámky	- Funkce předpokládá, že fonty "CZ-Regular" a "CZ-Bold" už jsou zaregistrované v ReportLab. - Zarovnání těla je nastavené na RIGHT od druhého sloupce (1,1) → (-1,-1), takže první sloupec zůstává defaultně (typicky vlevo) – to bývá vhodné pro textové labely.

Název funkce	<code>prepare_report</code>
Umístění	<code>learany.reporting.rules_report</code>
Účel funkce	Vygeneruje finální PDF report pomocí ReportLab (markdown-like text + několik tabulek). Report: (Volitelně) převede <code>`md_text`</code> do jednoduché struktury nadpisů/odrážek a vloží do <code>`story`</code>. Podporuje: - "# ", "## ", "### " jako Heading1/2/3, - "- " jako odrážku, - prázdný řádek jako Spacer. - Přidá sekce s tabulkami: - „Přehled predikční síly shluků“ (<code>`df_cluster`</code>), - „Přehled predikční síly pravidel shluků“ (<code>`df_rules`</code> ve vybraných sloupcích),

	- „Vybrané scénáře“ (`df_scenario` / scénáře), - „Přidaná klasifikační pravidla ve shlucích“ (`rules` ve vybraných sloupcích). Tabulky se renderují přes `df_to_wrapped_table`. Funkce zároveň: - zaregistruje fonty DejaVuSans (regular/bold) jako "CZ-Regular"/"CZ-Bold", - přepíše fonty ve standardních stylech (Normal/Title/Heading1-3), - vytvoří `SimpleDocTemplate` v landscape A4 a zavolá `doc.build(story)`.
Vstupní proměnné	output_pdf (str path-like): Cesta k výstupnímu PDF souboru. md_text (str None): Text ve zjednodušeném markdown formátu (nadpisy/odrážky). df_cluster (pandas.DataFrame None): Tabulka shluků pro vložení do reportu. df_rules (pandas.DataFrame None): Tabulka pravidel shluků pro vložení do reportu. df_scenario (pandas.DataFrame None): Tabulka scénářů pro vložení do reportu. rules (pandas.DataFrame None): Tabulka „přidaných“ klasifikačních pravidel v kontextech.
Výstupní proměnné	None: Funkce zapisuje PDF na disk do `output_pdf`.

Název funkce	pick_top_rules_text
Umístění	learany.reporting.flow_report
Účel funkce	Vybere top-k pravidel pro zadaný shluk a vrátí je jako seznam textů. Funkce filtruje `rules_df` na řádky, kde `cluster_target == cluster`, a poté pravidla seřadí podle dostupného skórovacího sloupce v tomto pořadí preference: 1) "score" 2) "dominance" 3) "quality_wracc" Pro text pravidla použije: - "rule_str_named" (pokud existuje; typicky už s nahrazenými Q-kódy), - jinak fallback na "rule_str".
Vstupní proměnné	rules_df (pandas.DataFrame): Tabulka pravidel (např. z `pick_divers_rules` / evaluace pravidel). Očekává sloupce "cluster_target" a alespoň jeden z ("rule_str_named", "rule_str"). cluster (int): ID shluku, pro který se mají vybrat pravidla (hodnota ve sloupci cluster_target).

	k (int, default=3): Kolik pravidel vrátit.
Výstupní proměnné	list[str]: Seznam textových reprezentací top-k pravidel. Pokud nejsou data, vrátí [].

Název funkce	make_params_fn
Umístění	learany.reporting.flow_report
Účel funkce	Vrátí callback `params_fn(t, c, ids_subset)`, který generuje krátký numerický profil uzlu/hrany. Vygenerovaný `params_fn`: - vezme podmnožinu `X_by_t[t]` podle `ids_subset`, - vezme numerické sloupce pro čas t z `numeric_cols_by_t`, - spočítá mediány numerických proměnných a vybere top 3 nejvyšší, - vrátí seznam vět ve formátu "Medián {col}: {val:.2f}".
Vstupní proměnné	X_by_t (dict[int, pandas.DataFrame]): Mapování časového řezu t -> feature matice X (index = ID). numeric_cols_by_t (dict[int, list[str]]): Mapování t -> seznam numerických sloupců, které se mají profilovat.
Výstupní proměnné	Callable[[int, int, Iterable], list[str]]: Funkce `params_fn(t, c, ids_subset)`. Parametr `c` je v podpisu pro konzistenci (v těle se nepoužívá).

Název funkce	nodes_from_flows_used
Umístění	learany.reporting.flow_report
Účel funkce	Vyextrahuje množinu uzlů (t, cluster) z tabulky přechodů. Z každé hrany (t, c_from -> c_to) přidá: - uzel (t, c_from), - uzel (t+1, c_to).
Vstupní proměnné	flows_used (pandas.DataFrame): Tabulka flows (očekává sloupce "t", "c_from", "c_to").
Výstupní proměnné	set[tuple[int,int]]: Unikátní uzly jako dvojice (t, cluster).

Název funkce	edges_from_flows_used
Umístění	learany.reporting.flow_report
Účel funkce	Vyextrahuje seznam hran (t, c_from, c_to) z tabulky přechodů.
Vstupní proměnné	flows_used (pandas.DataFrame): Tabulka flows (očekává sloupce "t", "c_from", "c_to").
Výstupní proměnné	list[tuple[int,int,int]]: Hran jako trojice (t, c_from, c_to) v pořadí řádků `flows_used`.

Název funkce	node_summary
Umístění	learany.reporting.flow_report
Účel funkce	Sestaví souhrn (metadatový slovník) pro uzel $N[t,c]$ = shluk c v čase t. Z `panel` vybere ID, které mají v `time_cols[t]` hodnotu `c`, a spočítá: - `n`: počet instancí v uzlu, - `share`: podíl vůči celé populaci v panelu ($n / \text{len}(\text{panel})$). Volitelně doplní: - `rules`: top pravidla pro daný shluk v čase t (přes `pick_top_rules_text`), - `params`: parametrický profil pro dané ID (přes `params_fn(t, c, ids)`).
Vstupní proměnné	panel (pandas.DataFrame): Wide tabulka s časovými sloupci shluků a identifikátorem `id_col`. id_col (str): Název sloupce s ID (např. "ID_STUDIA"). time_cols (list[str]): Seznam názvů sloupců s shluky v čase (např. ["cluster_t0", ...]). t (int): Index času (index do `time_cols`). c (int): ID shluku v čase t. rules_by_t (dict[int, pandas.DataFrame] None): Mapování $t \rightarrow$ tabulka pravidel pro daný čas. Očekává se formát kompatibilní s `pick_top_rules_text`. params_fn (callable None): Funkce `params_fn(t, c, ids_subset) \rightarrow list[str]` pro numerický profil. top_rules_k (int, default=5): Kolik pravidel max vrátit.
Výstupní proměnné	dict: Struktura uzlu např.: <pre> { "code": "N[t,c]", "t": t, "cluster": c, "n": n, "share": share, "rules": [...], "params": [...] } </pre>

Název funkce	edge_summary
Umístění	learany.reporting.flow_report
Účel funkce	Sestaví souhrn (metadatový slovník) pro hranu/přechod $E[t,a \rightarrow b]$ mezi časy t a $t+1$. Z `panel` vybere: - `ids_from`: ID ve zdrojovém uzlu (čas t, shluk c_{from}), - `ids_edge`: ID, která skutečně přešla (čas t: c_{from} AND čas $t+1$: c_{to}). Spočítá: - `n_from`: velikost zdrojového uzlu,

	- <code>`n_edge`</code>: velikost přechodu, - <code>`share_in_from`</code>: podíl přechodu v rámci zdroje (<code>n_edge / n_from</code>). Volitelně doplní: - <code>`params_from_on_edge_ids`</code>: parametrický profil pro přešlé (v čase <code>t</code>), - <code>`params_to_on_edge_ids`</code>: parametrický profil pro přešlé (v čase <code>t+1</code>), - <code>`rules_from`</code> / <code>`rules_to`</code>: top pravidla typická pro zdrojový/cílový uzel v daných časech.
Vstupní proměnné	<code>panel</code> (<code>pandas.DataFrame</code>): Wide tabulka s přiřazením shluků v čase. <code>id_col</code> (<code>str</code>): Název sloupce s ID. <code>time_cols</code> (<code>list[str]</code>): Časové sloupce s cluster assignmentem. <code>t</code> (<code>int</code>): Index přechodu (pracuje s <code>`time_cols[t]`</code> a <code>`time_cols[t+1]`</code>). <code>c_from</code> (<code>int</code>): Zdrojový shluk v čase <code>t</code>. <code>c_to</code> (<code>int</code>): Cílový shluk v čase <code>t+1</code>. <code>rules_by_t</code> (<code>dict[int, pandas.DataFrame] None</code>): Mapování <code>t</code> -> pravidla pro čas <code>t</code>. <code>params_fn</code> (<code>callable None</code>): Funkce <code>`params_fn(t, c, ids_subset)`</code> pro numerický profil. <code>top_rules_k</code> (<code>int</code>, <code>default=5</code>): Kolik pravidel max vrátit pro zdroj/cíl.
Výstupní proměnné	<code>dict</code> : Souhrn hrany s kódem " <code>E[t,a→b]</code> " a metrikami přechodu.

Název funkce	<code>render_node_md</code>
Umístění	<code>learany.reporting.flow_report</code>
Účel funkce	Vyrenderuje souhrn uzlu (z <code>`node_summary`</code>) do markdownu. Výstup obsahuje: - nadpis s kódem uzlu a časem, - počet a podíl, - volitelně sekci „Parametrický profil“ (prvních max ~6 položek), - volitelně sekci „Top pravidla“ (všechna předaná pravidla).
Vstupní proměnné	<code>ns</code> (<code>dict</code>): Slovník uzlu vrácený <code>`node_summary`</code> .
Výstupní proměnné	<code>str</code> : Markdown blok pro uzel.

Název funkce	<code>render_edge_md</code>
Umístění	<code>learany.reporting.flow_report</code>
Účel funkce	Vyrenderuje souhrn hrany (z <code>`edge_summary`</code>) do markdownu. Výstup obsahuje: - nadpis s kódem hrany a přechodem času, - definici přechodu mezi uzly,

	- počet a podíl přechodu ve zdrojovém uzlu, - volitelně profil přešlých ve zdroji i cíli (top ~5 vět), - volitelně „kotvící“ pravidla pro zdroj a cíl (top ~4).
Vstupní proměnné	es (dict): Slovník hrany vrácený `edge_summary`.
Výstupní proměnné	str: Markdown blok pro hranu.

Název funkce	build_atlas_report_markdown
Umístění	learany.reporting.flow_report
Účel funkce	Sestaví „atlas“ uzlů a přechodů do markdownu, konzistentní s právě použitými Sankey toky. Funkce: - vybere uzly a hrany pouze z `flows_used` (tedy jen to, co se skutečně kreslí / analyzuje), - pro každý uzel vytvoří `node_summary` a vyrenderuje jej přes `render_node_md`, - pro každou hranu vytvoří `edge_summary` a vyrenderuje ji přes `render_edge_md`, - doplní úvodní vysvětlivky („Jak číst“) a sekce Katalog uzlů / Katalog hran.
Vstupní proměnné	panel (pandas.DataFrame): Panel s časovými shluky (např. z `prepare_flow_panel`). id_col (str): Sloupec s ID (např. "ID_STUDIA"). time_cols (list[str]): Seznam časových sloupců (např. ["cluster_t0", ...]). flows_used (pandas.DataFrame): Tabulka přechodů použitá pro Sankey (t, c_from, c_to, count, ...). rules_by_t (dict[int, pandas.DataFrame] None): Mapování času na pravidla pro shluky v daném čase. params_fn (callable None): Funkce pro parametrické profily (`params_fn(t, c, ids_subset`).
Výstupní proměnné	str: Markdown dokument jako jeden string.

Název funkce	md_to_pdf_reportlab
Umístění	learany.reporting.flow_report
Účel funkce	Vygeneruje PDF report z jednoduchého markdown-like textu pomocí ReportLab, volitelně s obrázkem Sankey. Funkce: - zaregistruje fonty "CZ-Regular"/"CZ-Bold" (DejaVuSans) pro diakritiku, - nastaví je do základních ReportLab stylů, - pokud je `sankey` zadané, vloží do dokumentu nadpis a obrázek (Image),

	- převede `md_text` po řádcích do Paragraph/Spacer podle prefixů: "# " -> Heading1 "## " -> Heading2 "### " -> Heading3 "- " -> odrážka (•) jinak -> Normal text - uloží PDF do `output_pdf` (A4 portrait).
Vstupní proměnné	output_pdf (str path-like): Cesta k výstupnímu PDF. md_text (str None): Markdown-like text (nadpisy/odrážky). sankey (str path-like None): Cesta k obrázku Sankey (např. PNG), který se vloží do PDF.
Výstupní proměnné	None (ukládá soubor na uživatelem dané místo)

Seznam využívaných knihoven

pandas>=1.4

numpy~=1.26

scikit-learn~=1.6

scipy~=1.13

pysubgroup==0.8

matplotlib~=3.9

plotly~=5.24

reportlab~=4.4

Možnost instalace přes requirements.txt v balíčku:

```
pip install -r requirements.txt
```