Ranktable.py

Create rank table for the parameters → Fetch technical parameters and attributes within parameters along with their order/rank from mkp_api_response (DB) → Fetch the json_response value and load using 'json' module → **[meta] [result]=='success'** --False--> temp="Inactive Category"

[meta][result]=='success' --True-->

Iterate through each parameter and fetch specification details of attribute

Iterate through each 'specs' → Iterate through each 'sub_specs' → **sub_specs[is_golden]==True** → **sub_specs[dataType]=='groupedordinal'**

sub_specs[is_golden]==True ↓ **sub_specs[dataType]=='ordinal'**

Iterate through each sub_specs['set'] ←

Fetch and store the values of attributes
display_string=parameter_name
code=prameter_code
dtype=datatype
isfilter=isfilter
compare_criteria='No compare criteria'

sub_specs[dataType]=='ordinal' → Iterate through each sub_specs['set'] →

Fetch and store the values of attributes
display_string=parameter_name
code=parameter_code
dtype=datatype
isfilter=isfilter
compare_criteria='No compare criteria'
vname=parameter_attribute
val=value

sub_specs[dataType]=='groupedordinal' → Iterate through each sub_specs['set'] →

Fetch and store the values of attributes
display_string=parameter_name
code=parameter_code
dtype=datatype
isfilter=isfilter
compare_criteria='No compare criteria'
vname=parameter_attribute
val=value

→ Store the data to rank_data d/f

Calculate 'R score' for parameter

Iterate through rank_data d/f and calculate 'R score' based on value → Append the 'R score' to rank_data d/f → Calculate the 'R score' based on compare_criteria →

Rank Table <cat_id>.csv
[parameter,code,DataType,IsFilter,Compare Criteria, parameter_attribute,value,R score]

Getting_data.py

Fetch all the parameters from the Technical Parameter column which is in yaml format →

Read input files

Browse_node
Store the input_11.csv as dataframe ←

Code_to_parameter.csv
File is created manually with category_id,parameter and code
[Browse_node,Parameter,Parameter code]

Cat_specs
Store the Code_parameter.csv as dataframe ←

input11.csv
File is manually created with category_id's

Iterate over each category fetch product details from DB then preprocess on Technical_Parameters column

Iterate through each category ID in Browse_node → fetch product details for the category_id through a SQL query → Store DB o/p as 'data' d/f → Filter data for the specific category ID and active status → Offer_Price=Price-Discount Append offer_price column to data d/f

Concatenate the clean_para and 'data' d/f to 'final' d/f ← Store the details into clean_para d/f ← Extract relevant information from Technical_parameters by applying regular expression ← remove_map_lines('Technical_parameters') Removes lines starting with 'weight' from 'Technical Parameters'

Drop Technical_parameters column in 'final' → replace "NA"and "N/A" to "Not Applicable" → Rename column names from parameter code to parameter name using Cat_specs d/f in 'final' d/f → Append the offer_price to final d/f

store final data to <cat_id>.tsv
[make,model,Title,Product_id,Category_id,Category_name,
Discount,Price,offer_Price,<Technical parameter details>]

querytrans.py

Extract transaction data for the category_id → Fetch transaction data for the category_id through SQL query from DB → store data to <cat_id>.csv [variant_id,product_category_id, unit_price_value]

main.py
Creates score for each product

Iterate through each cat_id and perform preprocessing

Iterate through each category ID in Browse_node → Create cat_id folder → Create Filter.csv → Iterate through each category ID in Browse_node

Replace empty space with 'No brand' in brand column in 'data'

Store cat_id.tsv as 'data' d/f

Create importance table for each parameter: imp_table (data,filepath,cat_id)

Store <cat_id>.csv (Trnsction details) as df2 d/f → Groupby on variant_id and aggregate on unit_price_value and mean → Merge the data and df2 based on variant_id and product_id and store in df → Remove rows containing null values in df

Exception → Store the data in df

Create Randomforest model on (X = Parameters, y = mean_price) store it in model

Apply one-hot encoding on obj_list columns in df ← Select float and int datatype in df and store in num_list ← Select boolean datatype in df and store in obj_list

Create Randomforest model on (X = Parameters, y = offer_price) store it in model

Apply feature importance on model → Create a cdf_rf d/f with the parameters and their importance → Sort the parameters based on the importance value → Store the final data to Importance table.csv

Create rank table for each parameter attribute and remove columns which are not useful for the model

Call the rank_table(cat_id, filepath) in above Ranktable.py file → Store the data to rank_data d/f → Drop dulplicates in rank_data and store in rd → Fetch Data Type != groupedordinal,ordinal, manual,boolean & Compare Criteria != higher is better, lower is better in rd and store in col_list

removing useless parameters which are not useful for model building ← Remove the columns from col which are not in data and select those columns in data ← Append make, offer_price to col

Create rank table for each parameter attribute having No compare_criteria: coeff_table

```
┌──────────────────┐     ┌──────────────────┐     ┌──────────────────┐     ┌──────────────────────┐     ┌──────────────────┐
│  Store data in df│ ──▶ │ Store list of    │ ──▶ │ Select float and │ ──▶ │ Check if any numeric │ ──▶ │ Create dummies    │
│                  │     │ columns have     │     │ int datatype in  │     │ data having less than│     │ for object        │
│                  │     │ object datatypes │     │ df and store     │     │ 10 or 5% unique      │     │ data type columns │
│                  │     │ in obj_list      │     │ in num_list      │     │ of total records,    │     │                   │
│                  │     │                  │     │                  │     │ then it will be      │     │                   │
│                  │     │                  │     │                  │     │ considered as        │     │                   │
│                  │     │                  │     │                  │     │ categorical type     │     │                   │
└──────────────────┘     └──────────────────┘     └──────────────────┘     └──────────────────────┘     └──────────────────┘
```