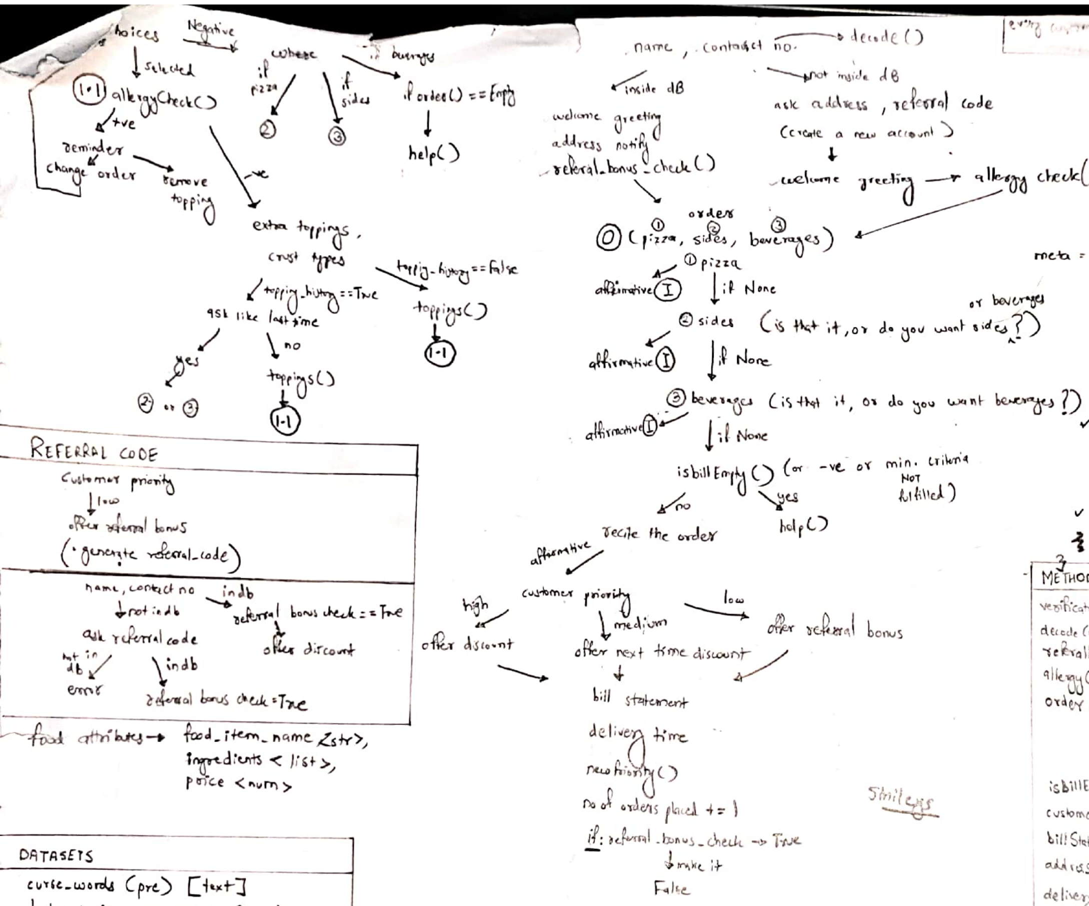

hoices
Negative
selected
1·1
allergyCheck()
+ve
reminder
change order
remove topping
-ve
extra toppings, crust types
topping_history==True
ask like last time
yes
② or ③
no
toppings()
1·1
topping_history==False
toppings()
1·1
where
if pizza
②
if sides
③
if beverages
if order() == Empty
help()
name, contact no.
decode()
inside dB
welcome greeting
address notify
referal_bonus_check()
not inside dB
ask address, referral code
(create a new account)
welcome greeting → allergy check(store)
every customer → unique PI key
order
⓪ (① pizza, ② sides, ③ beverages)
① pizza
affirmative Ⓘ
if None
② sides (is that it, or do you want sides or beverages?)
affirmative Ⓘ
if None
③ beverages (is that it, or do you want beverages?)
affirmative Ⓘ
if None
isbillEmpty() (or -ve or min. criteria NOT fulfilled)
no
yes
help()
recite the order
affirmative
customer priority
high
offer discount
medium
offer next time discount
low
offer referral bonus
bill statement
delivery time
newPriority()
no of orders placed += 1
if: referral_bonus_check → True
make it
False
Smileys
Synonym based universals
food_top_per_user
Wrong spelling!

REFERRAL CODE

Customer priority
↓ low
offer referral bonus
(· generate referral_code)

name, contact no → indb → referral bonus check == True → offer discount
↓ not indb
ask referral code
not in db → error
↓ indb
referral bonus check = True

food attributes → food_item_name <str>,
ingredients <list>,
price <num>

DATASETS

curse_words (pre) [text]
hot → pizza with toppings, → sides, → beverages } (grown) [csv]
priority (~~grown~~ pre) [csv]
comments → invalid input, → greetings, → apologize, → food quotes } (pre) [txt], → UNIVERSAL reply detection
meta-store (grown)
↳ contact no, referral bonus, etc. (entire meta) [sqlite]

priority	customer support
low	email based
medium	email based, immediate customer care when highly annoyed
high	customer care on slight annoyance

meta = { UID <primary key>
name <string>
contactno <encrypted>
address <string>
no_of_orders_placed <list of tuple>
order_worth <float>
discount_worth <float>
✓ allergies <list>
order_dates <list>
priority <not decided>
✓ topping_history <dict>
referral_bonus_check <bool>
referral_code <int> }

Customer annoyance level

base = 5 (for all priority)
help → (-2)
'did not understand you' → (-1)
change → (-1)
cancel → (-5)
curse words → (-4)
start over → (-3)

annoyance : <=1 (high)
<0 (medium)
<=-2 (low)

METHODS

verification()
decodeContactNo(encode <bool>)
referralBonusCheck()
allergyCheck(store <bool>)
order(type < 'pizza',
'sides',
'beverages' >)
isBillEmpty()
customerPriority()
billStatement()
addressInfo(store <bool>)
deliveryTime()
toppings(remove <bool>)
universalCheck()
annoyanceLevel()
Support()
easterEgg(;))
parser() (heart of methods)
userInput()
inputParse()

UNIVERSALS

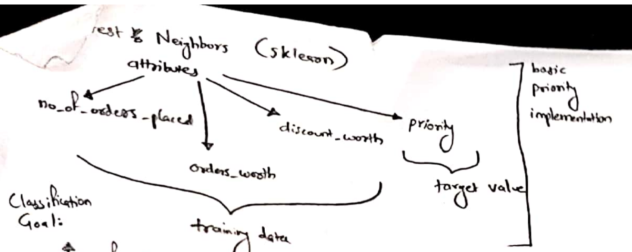
cancel (dynamic) ~~start over (dynamic)~~
top (dynamic) menu (comments)
hot (dynamic)
{curse_words}
help (comments)
about (comments)
change * (dynamic)
junk food, hungry, thirsty } (dynamic)

topping_history = { pizza <list>,
sides <list> }
↓
dips

no_of_orders = [(order_date,
order_worth),
⋮
]

change *
change my *
change the *
} to change details

rest K Neighbors (sklearn)


attributes
no_of_orders_placed
orders_worth
discount_worth
priority
training data
target value

basic
priority
implementation

Classification
Goal:

↑ no_of_orders_placed

↑ orders_worth

↓ discount_worth

topping history → we update : old_toppings or new_toppings } or } dict.
every time

Customers

- name and contact no. will be asked separately to reduce complexity
- contact no. will be encoded using SHA1 hash with first 10 characters stored in database
- check if contact no. is invalid by checking if its NUMERIC
- address will be broken down (split) in hopes of extracting the landmarks (eg:- Kandivali, borivali, andheri, etc.)
- address will be asked separately along with referral code (for not in db case)
- REMEMBER to store meta (SQL data) at the end, IN CASE user wants to cancel his/her order

enchant → To find spelling errors or nearest relevant word/s

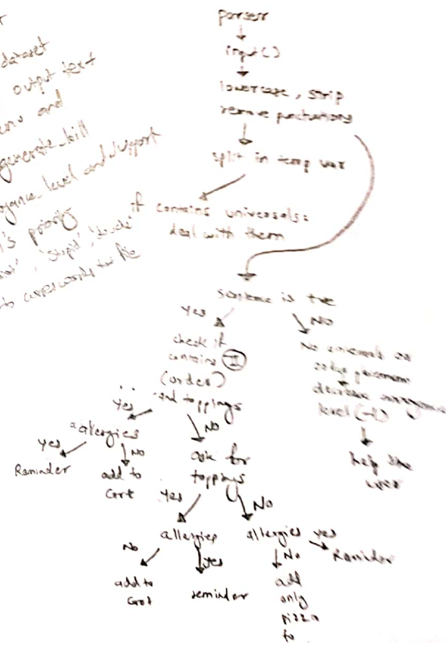
nltk → sent_tokenize
word_tokenize
stem.WordNetLemmatizer
↳ lemmatize( )

sklearn → feature_extraction.text.TfidfVectorizer
metrics.pairwise.cosine_similarity
neighbors.KNeighborsClassifier

dhruvam: • change order requirements.txt

devesh: • smileys in text dataset
• cleaning o/p output text & menu and generate_bill

me: • customer_annoyance_level and support
check sahil's priority
add 'idiot', 'stupid', 'dumb' to curse words file


parser
input()
lowercase, strip
remove punctuations
split in temp var
if contains universals:
deal with them
sentence is true
Yes
No
check if contains (II) (order) and toppings
Yes
allergies
Yes
Reminder
No
add to cart
No
ask for toppings
Yes
allergies
No
add to cart
Yes
reminder
No
allergies
Yes
Reminder
No
add only pizza to
No universals or order phrases
decrease annoyance level (-1)
help the user

Remember to add invalid command dataset

want *
want a *
get *
get a *
for *
for a *
place an order for *

want the *
get the *
for the *
order *
order a *
order the *

} check word with kleene star for ordering pizza, sides, beverages (II)

have *
have the *
have a *

one two three four five six seven eight nine } → pizzas

To check toppings ~~in~~built use → use with / with extra / with more

want * with *
want a * with *
⋮
for all of them

} Remember to remove 'for' from stopwords list

Can remove a, the from stopwords list