Project Report

Index

How to use the Web App?

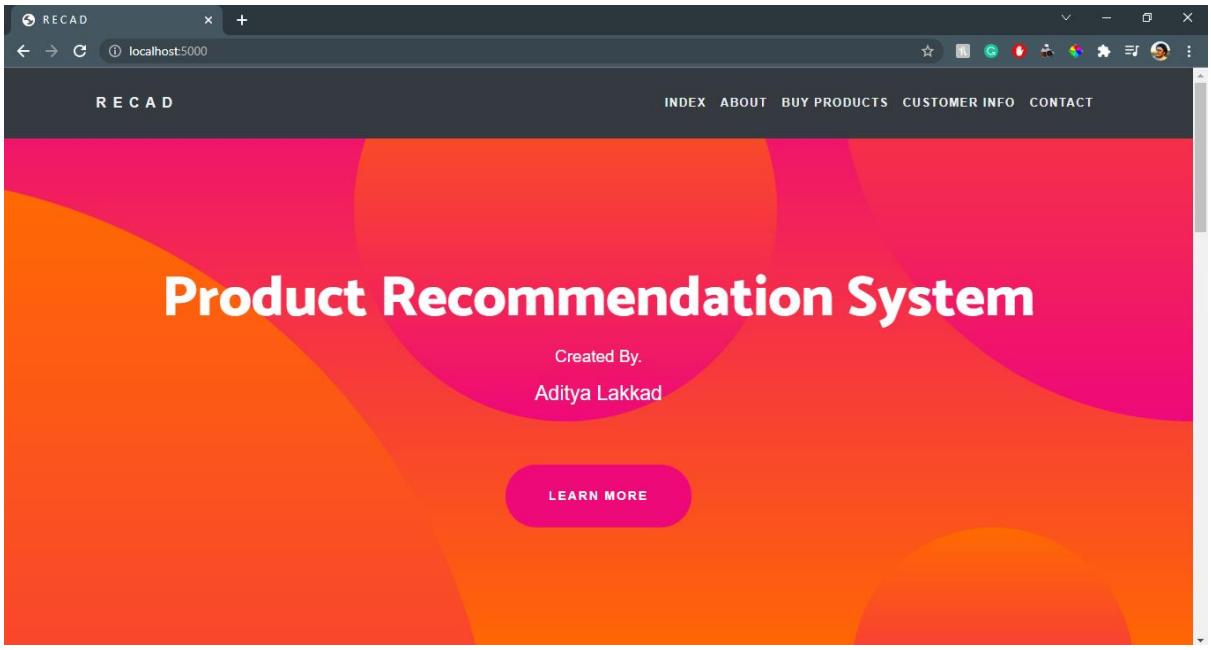
Step 1: First of all we have to run the project so run the command “python app.py” and wait for the response of the localhost URL.

```
PS E:\SEM-7\CGC\Project\Flask App> python app.py
Checking whether there is an H2O instance running at http://localhost:54321 ..
```

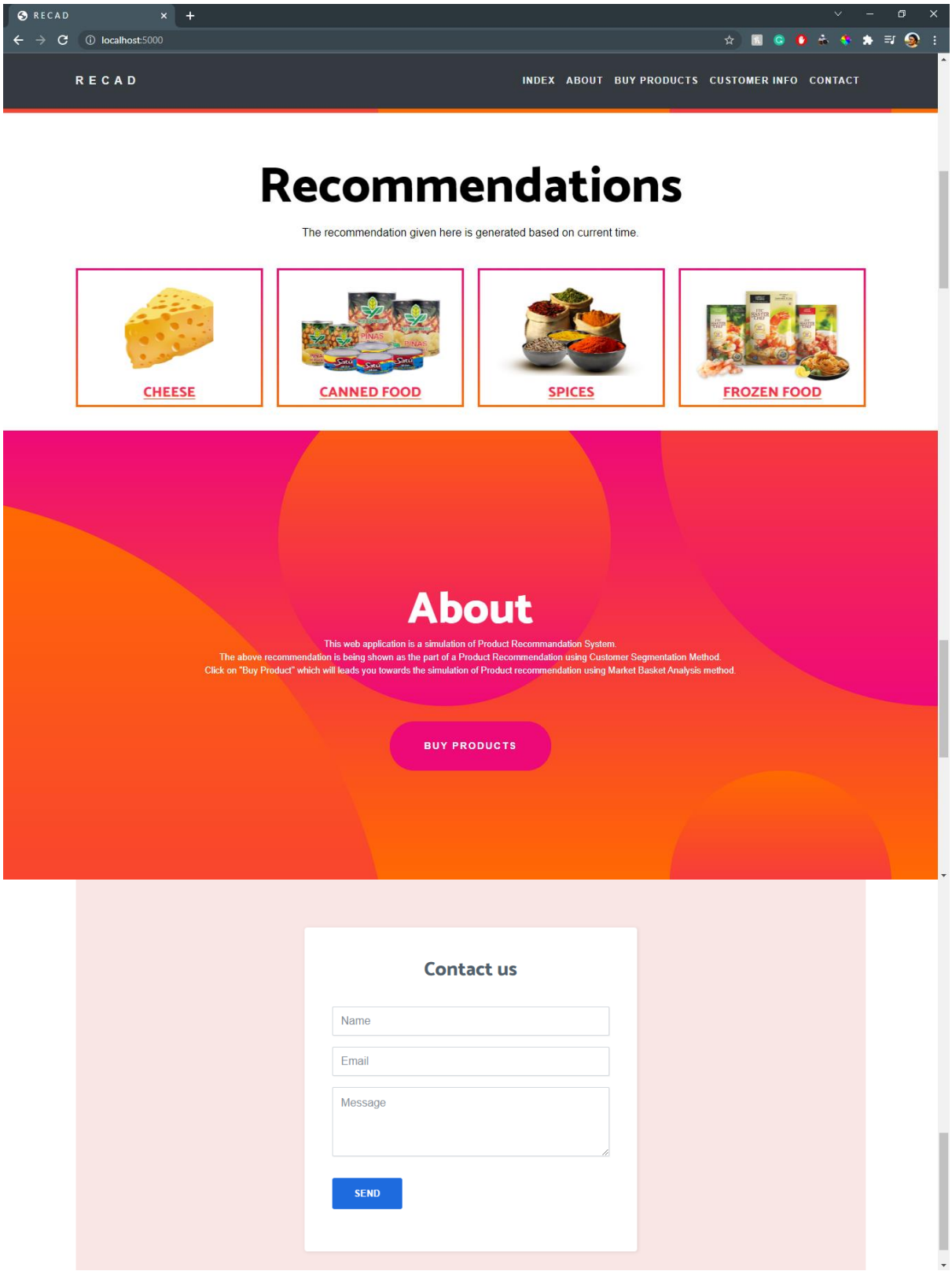
Step 2: Here we can see now our web app is up and running and here is a localhost URL that we can use to see our web app.

```
Python_version:             3.9.1 final
-------------------------   ------------------------------------------------------------
 * Debugger is active!
 * Debugger PIN: 228-784-400
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [22/Oct/2021 18:32:20] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [22/Oct/2021 18:32:22] "GET /favicon.ico HTTP/1.1" 404 -
```

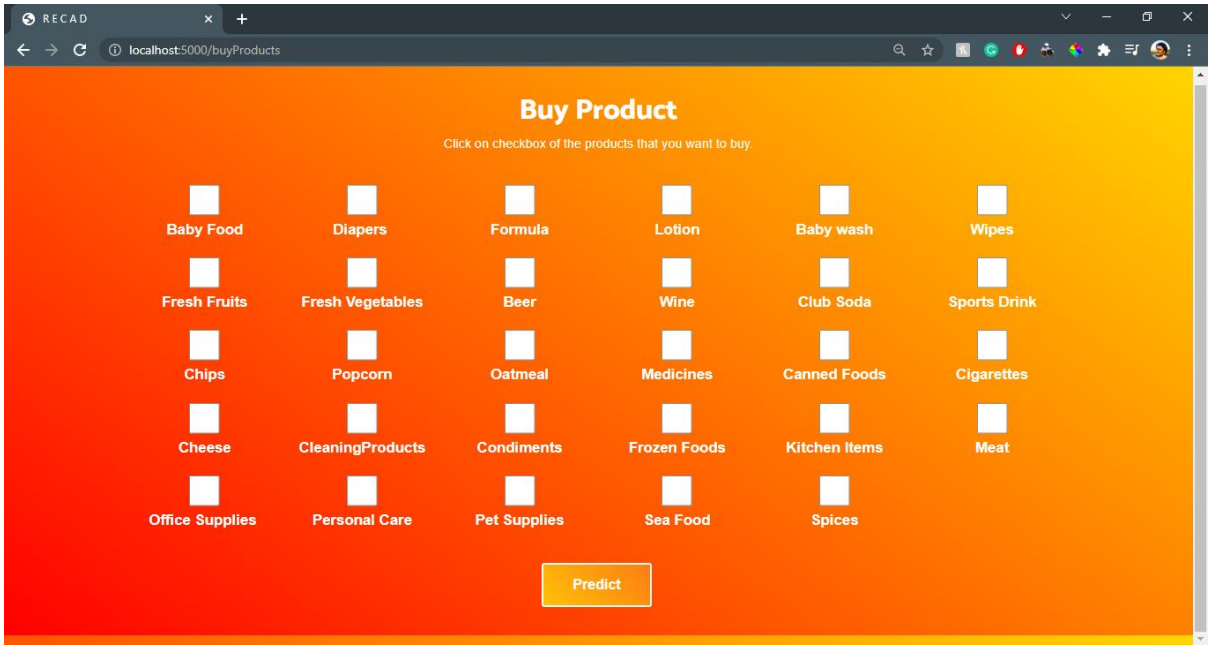
Step 3: Let’s open this URL. And here we have our webapp’s home page.



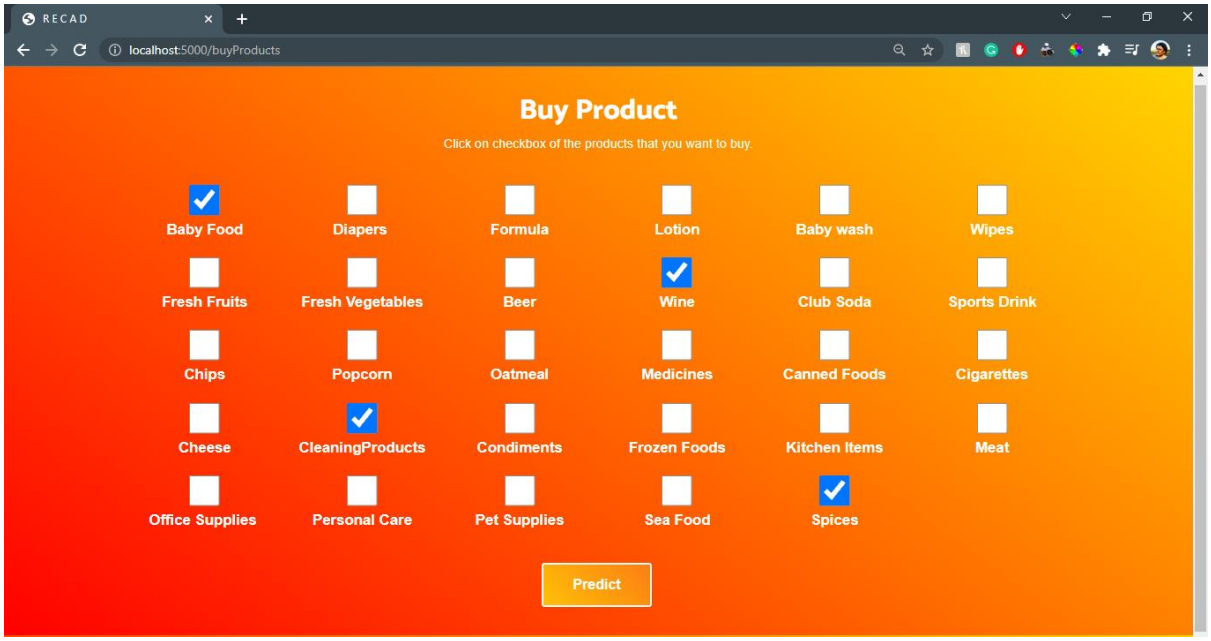
Step 4: You can see the time-based recommendation of the product and contact me form.



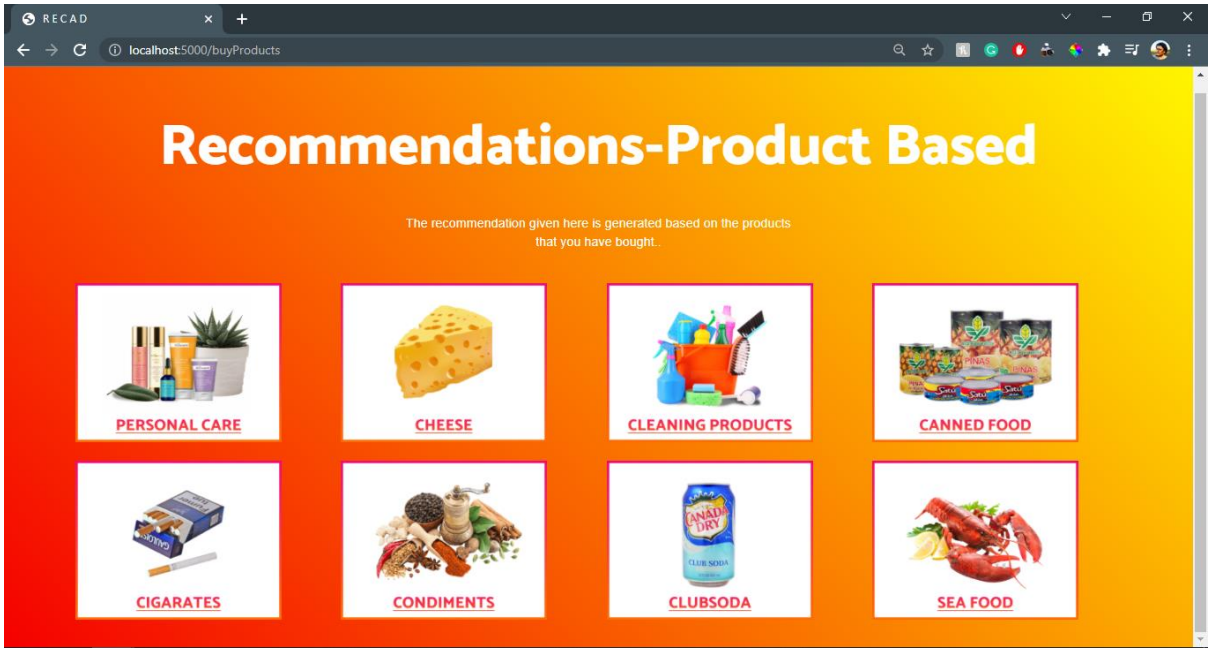
Step 5: Now click on “BUY PRODUCTS” to see the product purchase form.



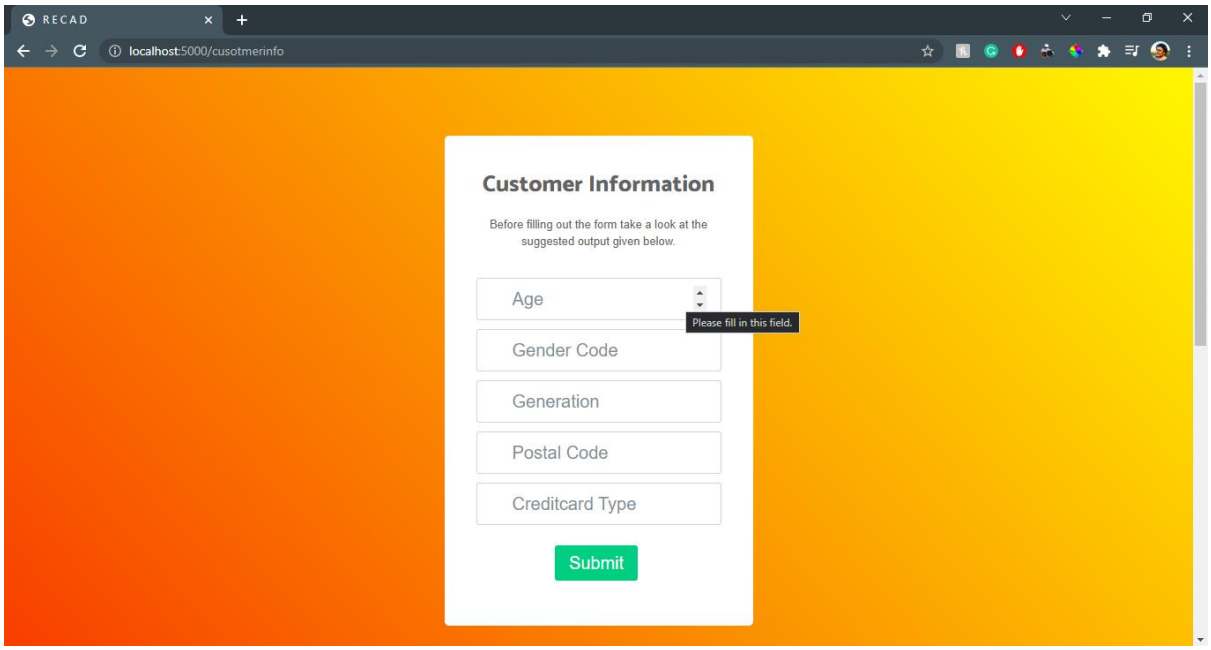
Step 6: Here I will select some of the product and click on the button “predict” given at last. This product will be sent to fetch results of association rule responses.



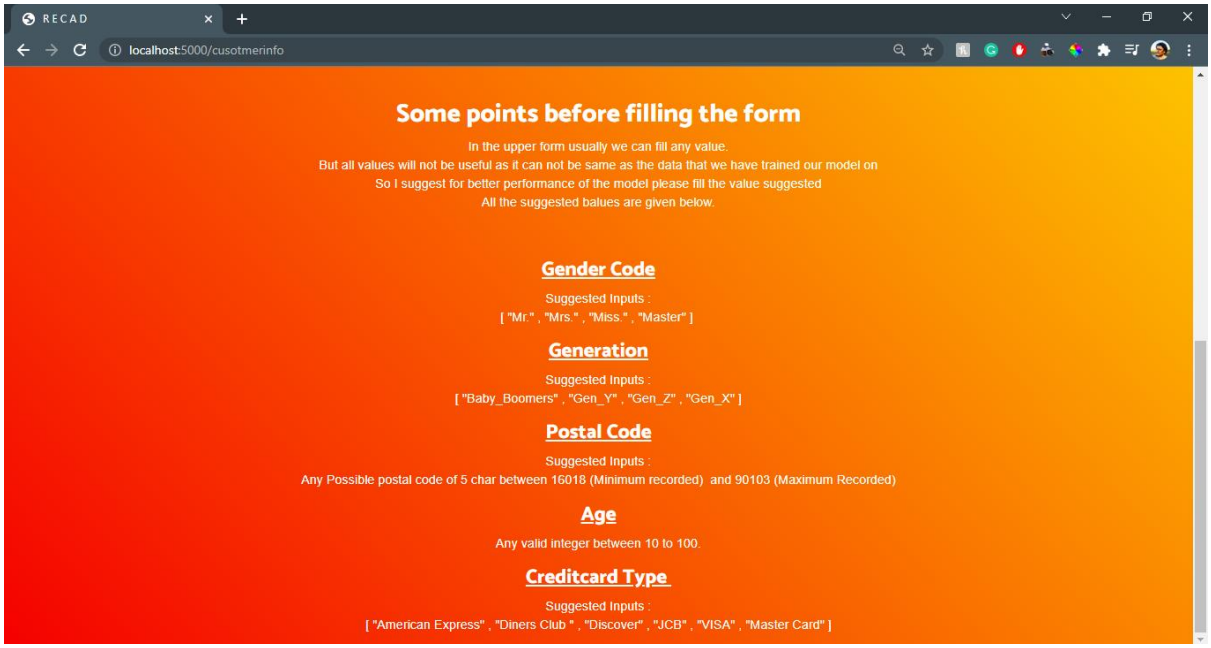
Step 7: Here we can see our recommended products from the associative rules.



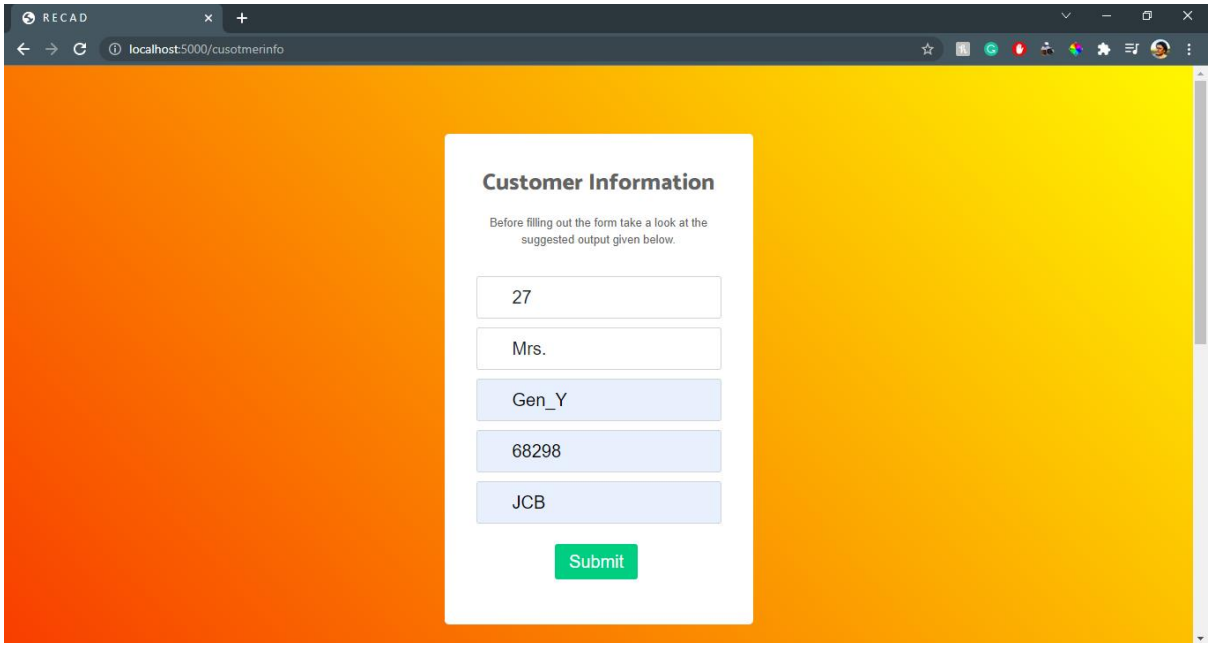
Step 8: Navigate towards to the home page and from there let's click on "CUSTOMER INFO". It's based on customer-based prediction. On this page here you can see a form.



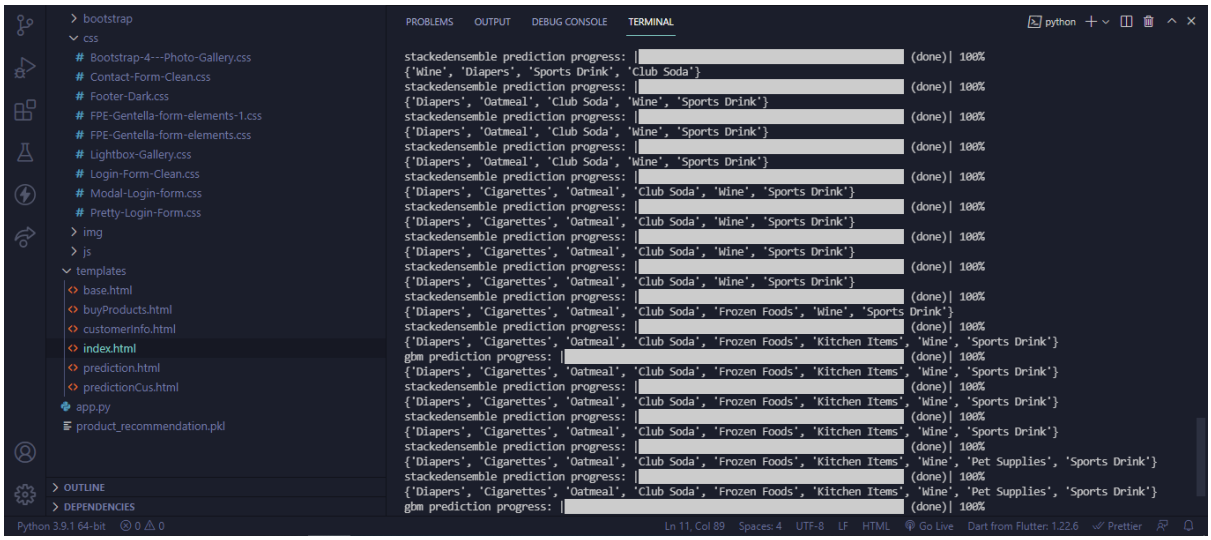
Step 9: Down below you can see the suggested input, giving suggested input will get you the best output.

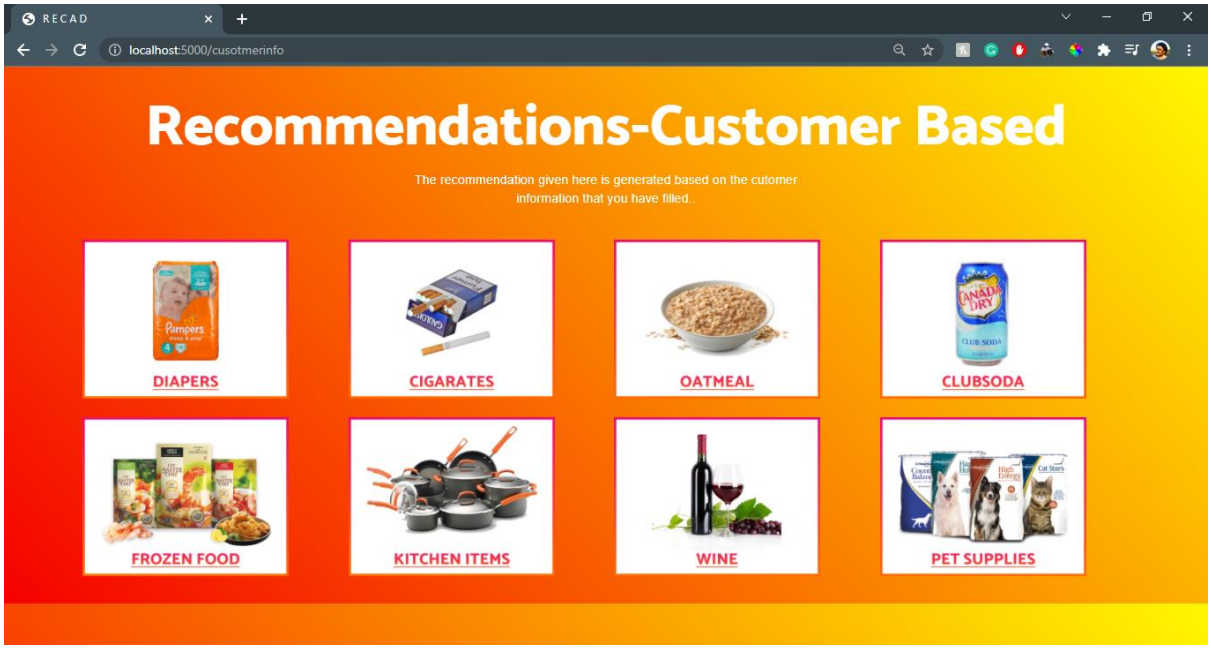


Step 10: Please fill those inputs in customer info form.



Step 11: Please wait for some time as there are models running in the backend for our customer info input. Based on that it will predict some items that we can see in the terminal. And it will be shown on the web page.





Explanation of Product Based Recommendation Notebook

Step 1: Loading the data.

```
In [1]: import pandas as pd

In [2]: data = pd.read_csv("order_transactions.csv")

In [3]: data.head()
```

Out[3]:

	CUSTNAME	GenderCode	ADDRESS1	CITY	STATE	COUNTRY_CODE	POSTAL_CODE	POSTAL_CODE_PLUS4	ADDRESS2	EMAIL_ADDRESS	...	C P
0	Allen Perl	Mr.	4707 Hillcrest Lane	Abeto	PG	IT	6040	0	NaN	Allen.M.Perl@spambob.com	...	
1	Allen Perl	Mr.	4707 Hillcrest Lane	Abeto	PG	IT	6040	0	NaN	Allen.M.Perl@spambob.com	...	
2	Allen Perl	Mr.	4707 Hillcrest Lane	Abeto	PG	IT	6040	0	NaN	Allen.M.Perl@spambob.com	...	
3	Allen Perl	Mr.	4707 Hillcrest Lane	Abeto	PG	IT	6040	0	NaN	Allen.M.Perl@spambob.com	...	
4	Allen Perl	Mr.	4707 Hillcrest Lane	Abeto	PG	IT	6040	0	NaN	Allen.M.Perl@spambob.com	...	

5 rows × 62 columns

Step 2: Now get all product column and store it in a dataframe.

```
In [4]: df_filtered = data[['CUST_ID','Baby Food', 'Diapers', 'Formula', 'Lotion', 'Baby wash', 'Wipes', 'Fresh Fruits', 'Fresh Vegetable

In [5]: df_customer_products = df_filtered.groupby('CUST_ID').sum()

In [6]: df_customer_products.head()
```

Out[6]:

CUST_ID	Baby Food	Diapers	Formula	Lotion	Baby wash	Wipes	Fresh Fruits	Fresh Vegetables	Beer	Wine	...	Cleaning Products	Condiments	Frozen Foods	Kitchen Items	Meat	Office Supplies	Personal Care
10001	0	0	1	0	0	1	0	1	0	1	...	1	0	0	0	0	0	0
10002	0	0	0	0	0	0	0	0	0	2	...	0	0	2	0	0	0	3
10003	0	1	2	2	1	0	0	1	1	0	...	2	1	0	0	0	0	0
10004	0	1	0	0	0	0	2	0	1	3	...	1	0	2	0	2	0	2
10005	1	0	0	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0

5 rows × 29 columns

Step 3: Encode the values as 0 for 0 and 1 for value greater then 1 or equals to 1. And also import libraries for apriori algo and association_rules.

```
In [7]: def encode_units(x):
            if x <= 0:
                return 0
            if x >= 0:
                return 1

        df_basket = df_customer_products.applymap(encode_units)

In [9]: from mlxtend.frequent_patterns import apriori
        from mlxtend.frequent_patterns import association_rules
```

Step 4: Get the frequent item sets from apriori algorithm with the support value. And using association rule get the confidence and lift.

```
In [10]: my_frequent_itemsets = apriori(df_basket, min_support=0.07, use_colnames=True)
         my_frequent_itemsets.head()
```

Out[10]:

	support	itemsets
0	0.140463	(Baby Food)
1	0.247613	(Diapers)
2	0.147889	(Formula)
3	0.133885	(Lotion)
4	0.141736	(Baby wash)

```
In [11]: my_rules = association_rules(my_frequent_itemsets, metric="lift", min_threshold=1)
         my_rules.head()
```

Out[11]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(Wine)	(Baby Food)	0.361341	0.140463	0.083174	0.230182	1.638743	0.032419	1.116546
1	(Baby Food)	(Wine)	0.140463	0.361341	0.083174	0.592145	1.638743	0.032419	1.565897
2	(Baby Food)	(Cigarettes)	0.140463	0.364099	0.084023	0.598187	1.642924	0.032881	1.582580
3	(Cigarettes)	(Baby Food)	0.364099	0.140463	0.084023	0.230769	1.642924	0.032881	1.117399
4	(Condiments)	(Baby Food)	0.543603	0.140463	0.077658	0.142857	1.017048	0.001302	1.002794

Step 5: Get the proper data frame value and store it using pickle.

```
my_rules[(my_rules['lift'] >= 3) & (my_rules['confidence'] >= 0.5)].sort_values(by='confidence', ascending=False)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
8	(Beer)	(Diapers)	0.146404	0.247613	0.146404	1.000000	4.038560	0.110152	inf
246	(Beer, Cigarettes)	(Diapers)	0.088479	0.247613	0.088479	1.000000	4.038560	0.066570	inf
4278	(Condiments, Meat)	(Spices, Cigarettes)	0.081052	0.298748	0.081052	1.000000	3.347301	0.056838	inf
258	(Spices, Beer)	(Diapers)	0.086145	0.247613	0.086145	1.000000	4.038560	0.064814	inf
1210	(Canned Foods, Pet Supplies)	(Cleaning Products)	0.074687	0.245703	0.074687	1.000000	4.069948	0.056336	inf
...	...	...	...	...	...	...	...	...	...
8122	(Sea Food, Personal Care)	(Spices, Wine, Cigarettes, Frozen Foods)	0.197751	0.154679	0.099512	0.503219	3.253320	0.068924	1.701597
7602	(Cheese, Wine, Frozen Foods)	(Spices, Medicines, Personal Care)	0.173138	0.154042	0.086993	0.502451	3.261779	0.060323	1.700251
9195	(Spices, Wine, Cigarettes, Medicines)	(Sea Food, Personal Care, Frozen Foods)	0.156588	0.144918	0.078294	0.500000	3.450220	0.055602	1.710163
9080	(Spices, Cheese, Personal Care)	(Wine, Sea Food, Cigarettes, Medicines)	0.155739	0.152132	0.077870	0.500000	3.286611	0.054177	1.695735
8411	(Spices, Cheese, Personal Care)	(Sea Food, Cigarettes, Medicines)	0.155739	0.152132	0.077870	0.500000	3.286611	0.054177	1.695735

Explanation of Customer Based Recommendation Notebook.

Step 1: Import all necessary libraries and initialize h2o.

```
In [1]: import pandas as pd
        import h2o
        from h2o.automl import H2OAutoML
        h2o.init()
```

```
Checking whether there is an H2O instance running at http://localhost:54321 ..... not found.
Attempting to start a local H2O server...
; Java HotSpot(TM) 64-Bit Server VM (build 25.311-b11, mixed mode)
  Starting server from C:\Users\Aditya\AppData\Local\Programs\Python\Python39\Lib\site-packages\h2o\backend\bin\h2o.jar
  Ice root: C:\Users\Aditya\AppData\Local\Temp\tmpqngfi9rg
  JVM stdout: C:\Users\Aditya\AppData\Local\Temp\tmpqngfi9rg\h2o_Aditya_started_from_python.out
  JVM stderr: C:\Users\Aditya\AppData\Local\Temp\tmpqngfi9rg\h2o_Aditya_started_from_python.err
  Server is running at http://127.0.0.1:54321
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
```

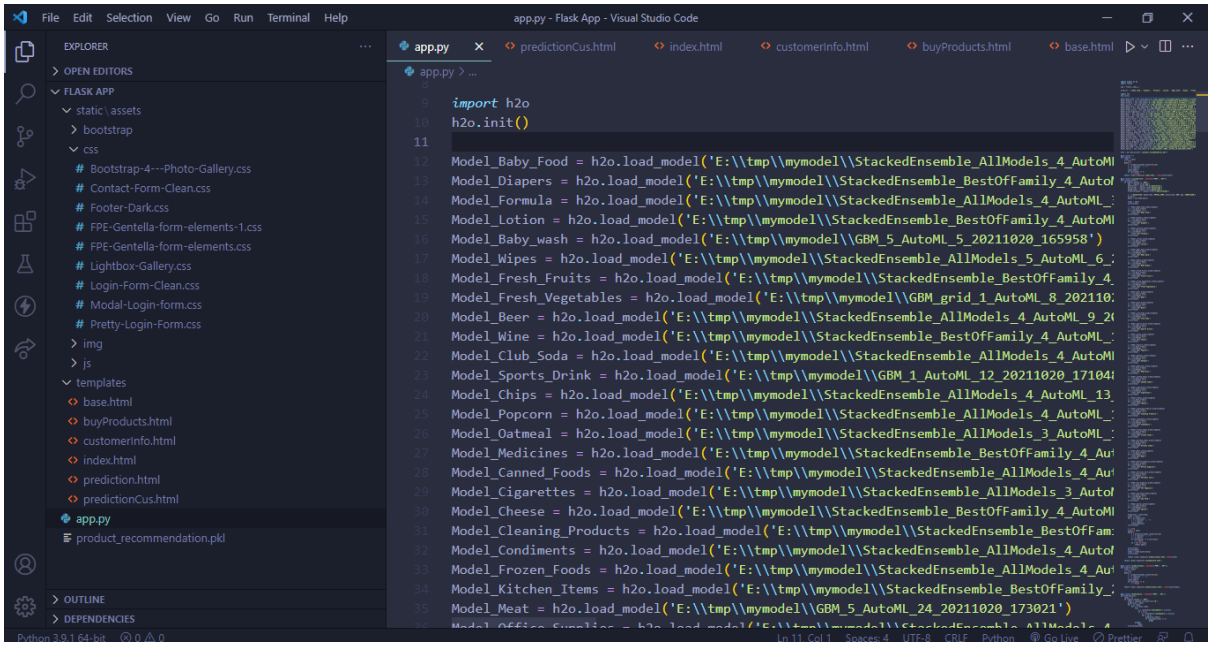
H2O_cluster_uptime:	07 secs
H2O_cluster_timezone:	Asia/Kolkata
H2O_data_parsing_timezone:	UTC
H2O_cluster_version:	3.34.0.3
H2O_cluster_version_age:	14 days, 23 hours and 58 minutes
H2O_cluster_name:	H2O_from_python_Aditya_jmh0jm
H2O_cluster_total_nodes:	1
H2O_cluster_free_memory:	1.746 Gb
H2O_cluster_total_cores:	4

Step 2: Get the products and create models for each product as well as train it on the customer data columns. And save the data files.

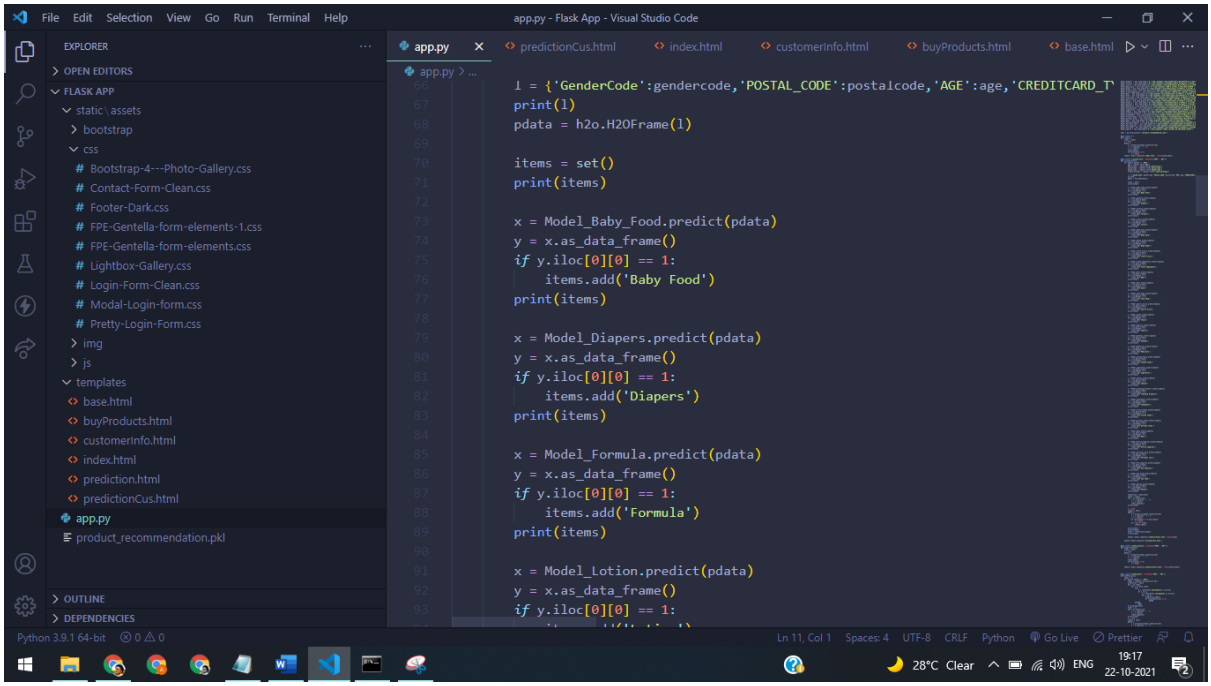
```
In [4]: x = ['GenderCode','POSTAL_CODE','AGE','CREDITCARD_TYPE','GENERATION']
        l = []
        model_path_dict = {}
        for y in product_list:
            df[y] = df[y].asfactor()
            aml = H2OAutoML(max_models = 10, seed = 1)
            aml.train(x = x, y = y, training_frame = df)
            l.append(aml.leaderboard.head(3))
            model = aml.leader
            model_path = h2o.save_model(model=model, path="/tmp/mymodel", force=True)
            model_path_dict[y] = model_path
```

```
AutoML progress: |
16:53:37.527: AutoML: XGBoost is not available; skipping it.
16:53:37.559: Step 'best_of_family_xgboost' not defined in provider 'StackedEnsemble': skipping it.
16:53:37.559: Step 'all_xgboost' not defined in provider 'StackedEnsemble': skipping it.

| (done) 100%
AutoML progress: |
16:55:31.612: AutoML: XGBoost is not available; skipping it.
16:55:31.612: Step 'best_of_family_xgboost' not defined in provider 'StackedEnsemble': skipping it.
16:55:31.612: Step 'all_xgboost' not defined in provider 'StackedEnsemble': skipping it.

| (done) 100%
AutoML progress: |
16:57:03.352: AutoML: XGBoost is not available; skipping it.
16:57:03.352: Step 'best_of_family_xgboost' not defined in provider 'StackedEnsemble': skipping it.
16:57:03.352: Step 'all_xgboost' not defined in provider 'StackedEnsemble': skipping it.

| (done) 100%
AutoML progress: |
```

Step 3: In the backend of the web app import all the files from the models that we have stored.



Step 4: After getting the data from the form use all this imported model to predict the values



And you can return all this values to the webapp for predictions.