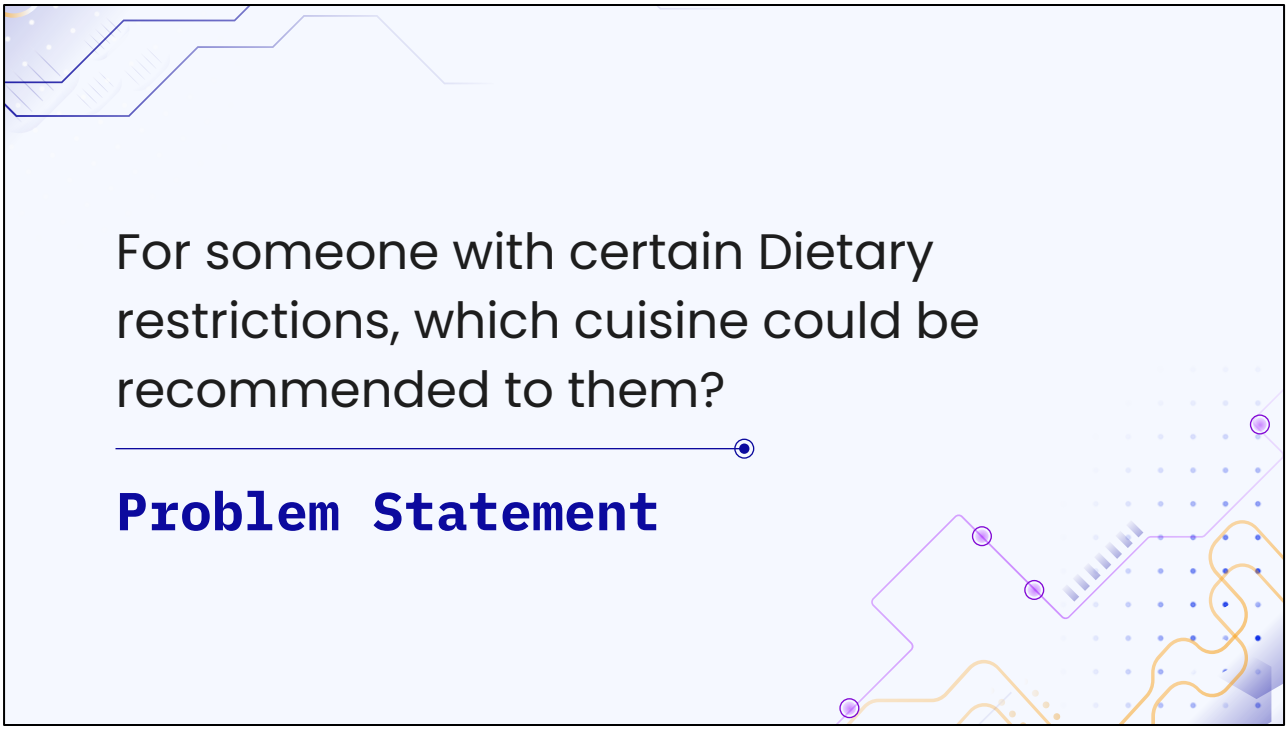
Good Afternoon Prof, I am Ilakkiyaa and these are my team members, Melvin, Samantha & Lily.
Today, we will be presenting on our Mini Project.

Table of contents

Before moving on, I would like to share the contents for today's presentation.
Firstly, i would be talking about our Problem Statement which is based on the "What's Cooking" Dataset.
Following that, Melvin would share the Exploratory Analysis done on the dataset which is basically, cleaning and preparing of the data together with the Visualisation.
Next up, Samantha will proceed to share details on the Machine learning section which discusses on the summary of the 3 models used as well as the Random Forest Model (Chosen)
Lastly, Lily will finish off the presentation with the outcome of the project as well as our contributions.

Now, I will be sharing the Problem Statement that we have chosen based on the “What’s Cooking” dataset.



Our Problem Statement for this project is:

For someone with certain Dietary restrictions, which cuisine could be recommended to them?

“What’s Cooking” Dataset

3 Variables, 39774 Entries	
Variable	**Description**
id	Recipe id
cuisine	Type of cuisine
ingredients	List of Ingredients of each recipe

Before proceeding to the next section, I would like to share the details from the dataset.
There are 3 variables; id, cuisine & ingredients
Id: shows the recipe id
Cuisine: Type of cuisine
Ingredients: List of ingredients of each recipe

Now, I would pass the time now to Melvin to explain more on the Exploratory Analysis done on our code.

Data Cleaning & Preparation

```
# Load and read the JSON train data file
cookingData = pd.read_json('train.json')
```

```
# Overview of the current cookingData
cookingData.head(10)
```

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
5	6602	jamaican	[plain flour, sugar, butter, eggs, fresh ginge...
6	42779	spanish	[olive oil, salt, medium shrimp, pepper, garli...
7	3735	italian	[sugar, pistachio nuts, white almond bark, flo...
8	16903	mexican	[olive oil, purple onion, fresh pineapple, por...
9	12734	italian	[chopped tomatoes, fresh basil, garlic, extra-...

The dataset from kaggle is presented in the .json format.

Firstly, a read json file is performed in jupyter to briefly look at the dataset.

The dataset from kaggle is presented in the json format. Firstly a read json file is performed in jupyter to briefly look at the dataset.

Data Cleaning & Preparation

```
# Convert lists to tuples in the 'ingredients' column
cookingData['ingredients'] = cookingData['ingredients'].apply(tuple)

# Drop duplicates
cookingData = cookingData.drop_duplicates()

# If needed, convert tuples back to lists
cookingData['ingredients'] = cookingData['ingredients'].apply(list)
```

```
cookingData.drop("id",axis=1)
```

	cuisine	ingredients
0	greek	[romaine lettuce, black olives, grape tomatoes...
1	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	indian	[water, vegetable oil, wheat, salt]
4	indian	[black pepper, shallots, cornflour, cayenne pe...
...	...	...
39769	irish	[light brown sugar, granulated sugar, butter, ...
39770	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	chinese	[boneless chicken skinless thigh, minced garli...
39773	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 2 columns

We then drop the duplicates under the ingredients column.

We then drop the duplicates under the ingredients column.

Data Cleaning & Preparation

```
json_data = pd.read_json('train.json')
json_data
```

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

From the above, we can see that the train data is presented as a json file format with 39774 rows and 3 columns.

The dataset from kaggle is presented in the .json format.

Firstly, a read json file is performed in jupyter to briefly look at the dataset.

Our best interest in this data cleaning step is to identify the list of **unique ingredients** and **cuisines**. Each list would then be written to a csv file and data frame for future references.

Data Cleaning & Preparation

```
#convert the json file into csv
json_data.to_csv('data_csv.csv', encoding='utf-8')

#import the csv file as a pandas dataframe
data_csv = pd.read_csv('data_csv.csv')

#rename columns properly
data_csv.columns = ["Index","ID","Cuisine","Ingredients"]

#store the renamed dataframe as a csv file
data_csv.to_csv('data_csv.csv', encoding='utf-8')
data_csv
```

	Index	ID	Cuisine	Ingredients
0	0	10259	greek	['romaine lettuce', 'black olives', 'grape tom...
1	1	25693	southern_us	['plain flour', 'ground pepper', 'salt', 'toma...
2	2	20130	filipino	['eggs', 'pepper', 'salt', 'mayonaise', 'cooki...
3	3	22213	indian	['water', 'vegetable oil', 'wheat', 'salt']
4	4	13162	indian	['black pepper', 'shallots', 'cornflour', 'cay...
...	...	...	...	...
39769	39769	29109	irish	['light brown sugar', 'granulated sugar', 'but...
39770	39770	11462	italian	['KRAFT Zesty Italian Dressing', 'purple onion...
39771	39771	2238	irish	['eggs', 'citrus fruit', 'raisins', 'sourdough...
39772	39772	41882	chinese	['boneless chicken skinless thigh', 'minced ga...
39773	39773	2362	mexican	['green chile', 'jalapeno chilies', 'onions', ...

39774 rows × 4 columns

First, we convert the given dataset from a .json file to a .csv file.

Data Cleaning & Preparation

	Index	ID	Cuisine	Ingredients
0	0	10259	greek	['romaine lettuce', 'black olives', 'grape tom...
1	1	25693	southern_us	['plain flour', 'ground pepper', 'salt', 'toma...
2	2	20130	filipino	['eggs', 'pepper', 'salt', 'mayonaise', 'cooki...
3	3	22213	indian	['water', 'vegetable oil', 'wheat', 'salt']
4	4	13162	indian	['black pepper', 'shallots', 'cornflour', 'cay...
...	...	...	...	...
39769	39769	29109	irish	['light brown sugar', 'granulated sugar', 'but...
39770	39770	11462	italian	['KRAFT Zesty Italian Dressing', 'purple onion...
39771	39771	2238	irish	['eggs', 'citrus fruit', 'raisins', 'sourdough...
39772	39772	41882	chinese	['boneless chicken skinless thigh', 'minced ga...
39773	39773	2362	mexican	['green chile', 'jalapeno chilies', 'onions', ...

39774 rows × 4 columns

We can see that under the **ingredients column**, it is currently treated as a string with square brackets and single quotes, hence we need to split up each of the ingredients from each cell.

Data Cleaning & Preparation

```
#removal process
for x in test_csv.Index:
    test_csv.Ingredients[x] = test_csv.Ingredients[x].replace("'",'').strip('[]')
test_csv
```

```
C:\Users\tanma\AppData\Local\Temp\ipykernel_7060\2268666816.py:3: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
w-versus-a-copy
  test_csv.Ingredients[x] = test_csv.Ingredients[x].replace("'",'').strip('[]')
```

	Index	ID	Cuisine	Ingredients
0	0	10259	greek	romaine lettuce, black olives, grape tomatoes,...
1	1	25693	southern_us	plain flour, ground pepper, salt, tomatoes, gr...
2	2	20130	filipino	eggs, pepper, salt, mayonaise, cooking oil, gr...
3	3	22213	indian	water, vegetable oil, wheat, salt
4	4	13162	indian	black pepper, shallots, cornflour, cayenne pep...
...	...	...	...	...
39769	39769	29109	irish	light brown sugar, granulated sugar, butter, w...
39770	39770	11462	italian	KRAFT Zesty Italian Dressing, purple onion, br...
39771	39771	2238	irish	eggs, citrus fruit, raisins, sourdough starter...
39772	39772	41882	chinese	boneless chicken skinless thigh, minced garlic...
39773	39773	2362	mexican	green chile, jalapeno chilies, onions, ground ...

39774 rows × 4 columns

Overwrites the cell with new info.

1. For each index x, it replaces single quotes with an empty string, and then remove square brackets from the beginning and end of the string.
2. The modified value is then assigned back to the "Ingredients" column at index x.

Data Cleaning & Preparation

	Index	ID	Cuisine	Ingredients
0	0	10259	greek	romaine lettuce, black olives, grape tomatoes,...
1	1	25693	southern_us	plain flour, ground pepper, salt, tomatoes, gr...
2	2	20130	filipino	eggs, pepper, salt, mayonaise, cooking oil, gr...
3	3	22213	indian	water, vegetable oil, wheat, salt
4	4	13162	indian	black pepper, shallots, cornflour, cayenne pep...
...	...	...	...	...
39769	39769	29109	irish	light brown sugar, granulated sugar, butter, w...
39770	39770	11462	italian	KRAFT Zesty Italian Dressing, purple onion, br...
39771	39771	2238	irish	eggs, citrus fruit, raisins, sourdough starter...
39772	39772	41882	chinese	boneless chicken skinless thigh, minced garlic...
39773	39773	2362	mexican	green chile, jalapeno chilies, onions, ground ...

Now, all ingredients do not have quotations and square brackets.

Data Cleaning & Preparation

```
#compilation of all ingredients.
ingredient_list = ""
for x in data_csv.Index:
    ingredient_list = ingredient_list+data_csv.Ingredients[x]+', '
ingredient_list
```

```
'romaine lettuce, black olives, grape tomatoes, garlic, pepper, purp
bles, plain flour, ground pepper, salt, tomatoes, ground black peppe
k, vegetable oil, eggs, pepper, salt, mayonaise, cooking oil, green
ow onion, soy sauce, butter, chicken livers, water, vegetable oil, v
e pepper, onions, garlic paste, milk, butter, salt, lemon juice, wat
s chicken skinless thigh, garam masala, double cream, natural yogurt
inger root, salt, ground cinnamon, milk, vanilla extract, ground gin
medium shrimp, pepper, garlic, chopped cilantro, jalapeno chilies, f
t, bay leaf, chorizo sausage, sugar, pistachio nuts, white almond ba
t, eggs, baking powder, dried cranberries, olive oil, purple onion,
s, cheddar cheese, ground black pepper, salt, iceberg lettuce, lime,
matoes, fresh basil, garlic, extra-virgin olive oil, kosher salt, f
o, olive oil, garlic, sharp cheddar cheese, pepper, swiss cheese, pr
sausages, low sodium soy sauce, fresh ginger, dry mustard, green bea
sugar, Shaoxing wine, garlic, ground turkey, water, crushed red pepp
ts, hot red pepper flakes, extra-virgin olive oil, fresh lemon juice
ea salt, flat leaf parsley, ground cinnamon, fresh cilantro, chili p
pper, garlic, plum tomatoes, avocado, lime juice, flank steak, salt,
epper flakes, onions, fresh parmesan cheese, butter, all-purpose fla
```

A similar command is used to pass each and every non unique ingredients into a string with commas being the separator.

```
#change the string of ingredients into a list instead
ingredient_list_non_unique = list(ingredient_list.split(", "))
ingredient_list_non_unique
```

```
['romaine lettuce',
 'black olives',
 'grape tomatoes',
 'garlic',
 'pepper',
 'purple onion',
 'seasoning',
 'garbanzo beans',
 'feta cheese crumbles',
 'plain flour',
 'ground pepper',
 'salt',
 'tomatoes',
 'ground black pepper',
 'thyme',
 'eggs',
 'green tomatoes',
 'yellow corn meal',
 'milk',
```

This string is then passed on to a list, which splits them based on the location of the commas.

Data Cleaning - Findings

```
#change the list of ingredients into a set because set cannot have duplicate values
ingredients_set = set(ingredient_list_non_unique)
final_ingredient_list = list(ingredients_set)
final_ingredient_list
```

```
['',
 'low-fat cream cheese',
 'Japanese mountain yam',
 'piri-piri sauce',
 'sliced fresh fruit',
 'rib roast',
```



```
final_ingredient_list.remove('') #the first data point is removed
number = len(final_ingredient_list)
print('Number of unique ingredients are: '+str(number))
final_ingredient_list
```

```
Number of unique ingredients are: 6724

['low-fat cream cheese',
 'Japanese mountain yam',
 'piri-piri sauce',
 'sliced fresh fruit',
 'rib roast',
 'Kewpie Mayonnaise',
 'paprika paste',
```

Next, we change the list of ingredients into a set.

Next, we remove the first null data point and find out the total number of **unique ingredients** are **6724**.

Data Cleaning - Findings

```
#change the list of ingredients into a set because set cannot have duplicate values
ingredients_set = set(ingredient_list_non_unique)
final_ingredient_list = list(ingredients_set)
final_ingredient_list
```

```
['',
 'low-fat cream cheese',
 'Japanese mountain yam',
 'piri-piri sauce',
 'sliced fresh fruit',
 'rib roast',
```



```
final_ingredient_list.remove('') #the first data point is removed
number = len(final_ingredient_list)
print('Number of unique ingredients are: '+str(number))
final_ingredient_list
```

```
Number of unique ingredients are: 6724

['low-fat cream cheese',
 'Japanese mountain yam',
 'piri-piri sauce',
 'sliced fresh fruit',
 'rib roast',
 'Kewpie Mayonnaise',
 'paprika paste',
```

Since sets in python cannot have duplicate values, we can change the list of ingredients into a set, duplicate elements will be removed, leaving only the **unique** ones.

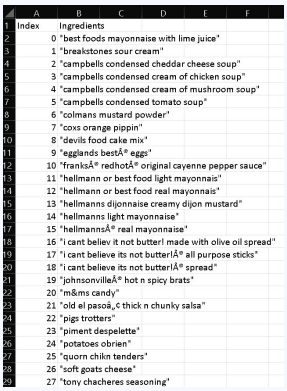
Next, we remove the first null data point and find out the total number of **unique ingredients** are **6724**.

Data Cleaning - Findings

```
#create a new dataframe consisting of ingredients only
ingredients_df = pd.DataFrame(final_ingredient_list)

#convert to csv file
ingredients_df.to_csv('ingredients.csv',index=False, encoding='utf-8')
ingredients_df = pd.read_csv('ingredients.csv')
ingredients_df.columns=['Ingredients'] #name empty columns
ingredients_df['Ingredients'] = ingredients_df['Ingredients'].str.capitalize()
ingredients_df=ingredients_df.sort_values(by="Ingredients")

#drop the old index and relabel the index column, starting from 0 onwards
ingredients_df=ingredients_df.reset_index(drop=True)
ingredients_df.to_csv('ingredients.csv', index = True, encoding='utf-8')
ingredients_df = pd.read_csv('ingredients.csv')

#rename the index column
ingredients_df.columns=['Index','Ingredients']
ingredients_df.to_csv('ingredients.csv', index = False, encoding='utf-8')
ingredients_df
```

	Index	Ingredients
0	0	"best foods mayonnaise with lime juice"
1	1	"breakstones sour cream"
2	2	"campbells condensed cheddar cheese soup"
3	3	"campbells condensed cream of chicken soup"
4	4	"campbells condensed cream of mushroom soup"
...	...	...
6719	6719	Zesty italian dressing
6720	6720	Zinfandel
6721	6721	Ziti
6722	6722	Zucchini
6723	6723	Zucchini blossoms

We then write our set of unique ingredients to a **csv** file for future reference.



	A	B
1	Index	Ingredients
2	0	"best foods mayonnaise with lime juice"
3	1	"breakstones sour cream"
4	2	"campbells condensed cheddar cheese soup"
5	3	"campbells condensed cream of chicken soup"
6	4	"campbells condensed cream of mushroom soup"
7	5	"campbells condensed tomato soup"
8	6	"colmans mustard powder"
9	7	"coxs orange pippin"
10	8	"devils food cake mix"
11	9	"egglands bestÂ® eggs"
12	10	"franksÂ® redhotÂ® original cayenne pepper sauce"
13	11	"hellmann or best food light mayonnais"
14	12	"hellmann or best food real mayonnais"
15	13	"hellmanns dijonnaise creamy dijon mustard"
16	14	"hellmanns light mayonnaise"
17	15	"hellmannsÂ® real mayonnaise"
18	16	"i cant believ it not butter! made with olive oil spread"
19	17	"i cant believe its not butterlÂ® all purpose sticks"
20	18	"i cant believe its not butterlÂ® spread"
21	19	"johnsonvilleÂ® hot n spicy brats"
22	20	"m&ms candy"
23	21	"old el pasoâ„¢ thick n chunky salsa"
24	22	"pigs trotters"
25	23	"piment despelette"
26	24	"potatoes obrien"
27	25	"quorn chikn tenders"
28	26	"soft goats cheese"
29	27	"tony chacheres seasoning"

Data Cleaning - Findings

	Index	Ingredients
0	0	"best foods mayonnaise with lime juice"
1	1	"breakstones sour cream"
2	2	"campbells condensed cheddar cheese soup"
3	3	"campbells condensed cream of chicken soup"
4	4	"campbells condensed cream of mushroom soup"
...	...	...
6719	6719	Zesty italian dressing
6720	6720	Zinfandel
6721	6721	Ziti
6722	6722	Zucchini
6723	6723	Zucchini blossoms

6724 rows × 2 columns

We then write our set of unique ingredients to a **csv** file for future reference.

Data Cleaning - Findings

	A	B	C
1	Index	Cuisines	
2	0	Brazilian	
3	1	British	
4	2	Cajun_creole	
5	3	Chinese	
6	4	Filipino	
7	5	French	
8	6	Greek	
9	7	Indian	
10	8	Irish	
11	9	Italian	
12	10	Jamaican	
13	11	Japanese	
14	12	Korean	
15	13	Mexican	
16	14	Moroccan	
17	15	Russian	
18	16	Southern_us	
19	17	Spanish	
20	18	Thai	
21	19	Vietnamese	
22			

The same steps are also used to sort out the cuisines uniquely.

We find out the number of **unique cuisines** are **20**.

Data Cleaning - Findings

	Index	Cuisines
0	0	Brazilian
1	1	British
2	2	Cajun_creole
3	3	Chinese
4	4	Filipino
5	5	French
6	6	Greek
7	7	Indian
8	8	Irish
9	9	Italian
10	10	Jamaican
11	11	Japanese
12	12	Korean
13	13	Mexican
14	14	Moroccan
15	15	Russian
16	16	Southern_us
17	17	Spanish
18	18	Thai
19	19	Vietnamese

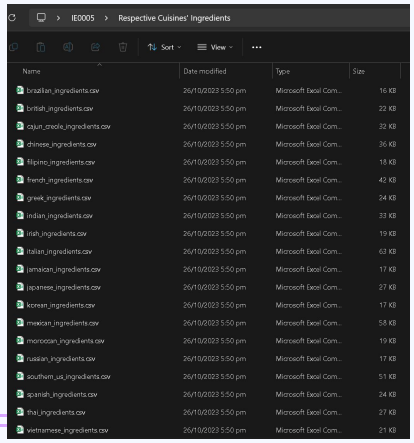
The same steps are also used to sort out the cuisines uniquely.

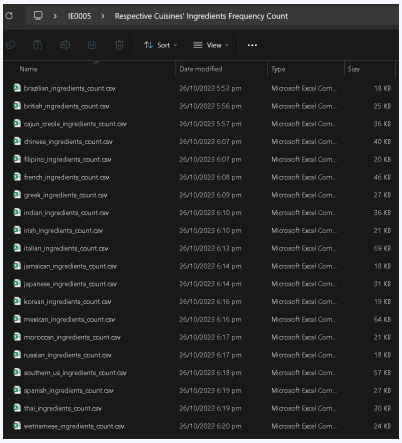
We find out the number of **unique cuisines** are **20**.

Data Cleaning - Results

Respective cuisines' ingredients and **Respective cuisines' ingredients frequency count** are also sorted out.

End result is this collection of csv files of cleaned data.

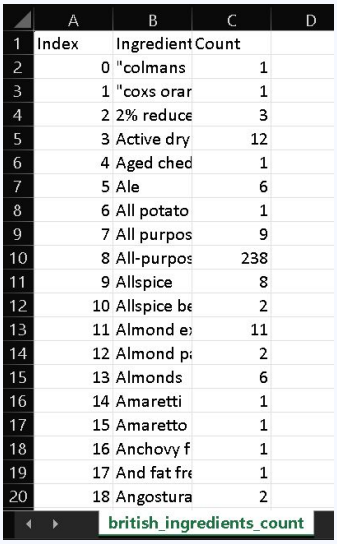




Data Cleaning - Results

Index	Ingredients
0	"colmans mustard powder"
1	"coxs orange pippin"
2	2% reduced-fat milk
3	Active dry yeast
4	Aged cheddar cheese
5	Ale
6	All potato purpos
7	All purpose unbleached flour
8	All-purpose flour
9	Allspice
10	Allspice berries
11	Almond extract
12	Almond paste
13	Almonds
14	Amaretti
15	Amaretto
16	Anchovy fillets
17	And fat free half half
18	Angostura bitters
19	Apple cider vinegar
20	Apple pie spice

british_ingredients

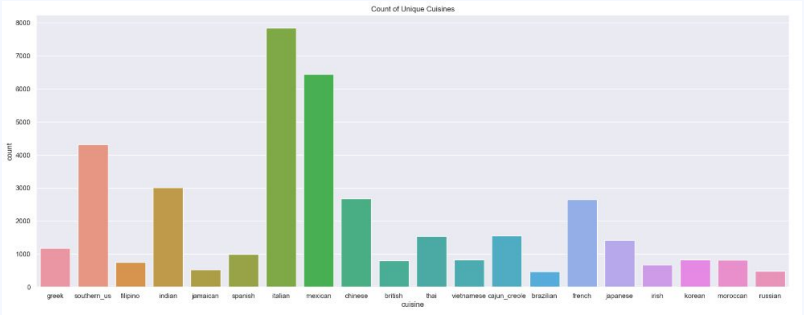


Index	Ingredient	Count
0	"colmans	1
1	"coxs orar	1
2	2% reduce	3
3	Active dry	12
4	Aged ched	1
5	Ale	6
6	All potato	1
7	All purpos	9
8	All-purpos	238
9	Allspice	8
10	Allspice be	2
11	Almond ex	11
12	Almond pa	2
13	Almonds	6
14	Amaretti	1
15	Amaretto	1
16	Anchovy f	1
17	And fat fre	1
18	Angostura	2

british_ingredients_count

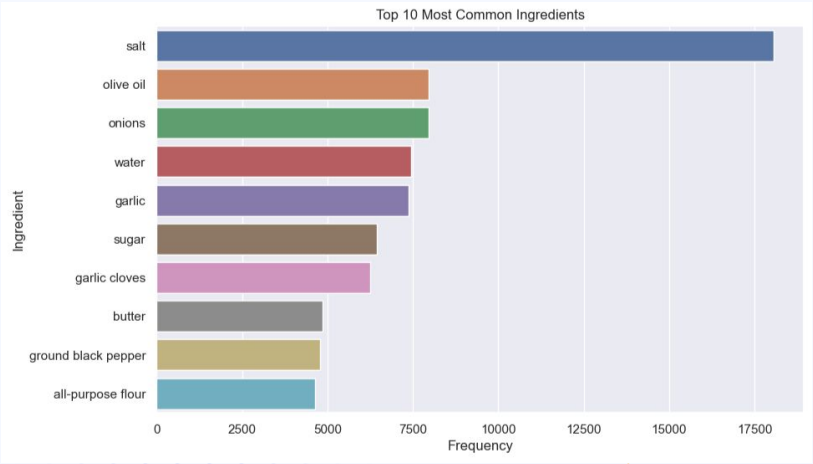
Here is an example of British ingredients (**unique**) and their frequency count.

Exploratory Analysis



Out of all the cuisine, most of the recipes came from **Italian** cuisine

Exploratory Analysis



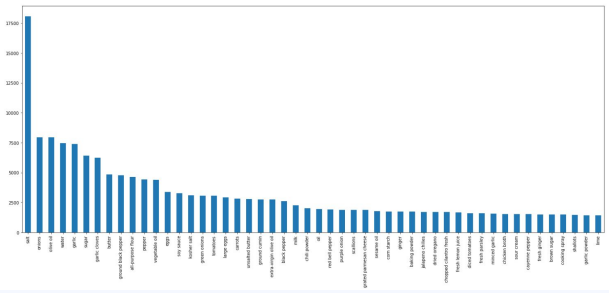
As plotted in the graph, across all the different cuisine types, the top 10 most common ingredients are shown in the graph

Exploratory Analysis

Data Visualisation can be done via catplots.
The following shows the most used ingredients overall and for each cuisine.

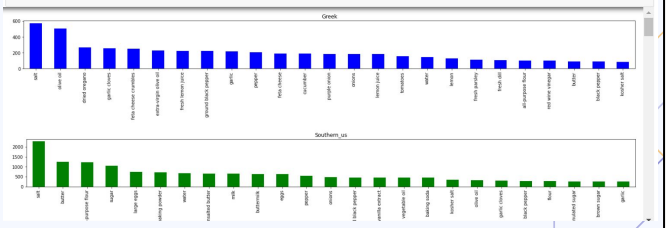
```
#we want to show the most commonly used ingredients among all the cuisines
fig, ax = plt.subplots(figsize=(25,10))
ingredients_count = []
for x in data_json['ingredients']:
    for y in x:
        ingredients_count.append(y)

ingredients_count = pd.Series(ingredients_count)
ingredients_count.value_counts().sort_values(ascending=False).head(50).plot.bar(ax = ax)
```

```
<AxesSubplot:>
```



Overall

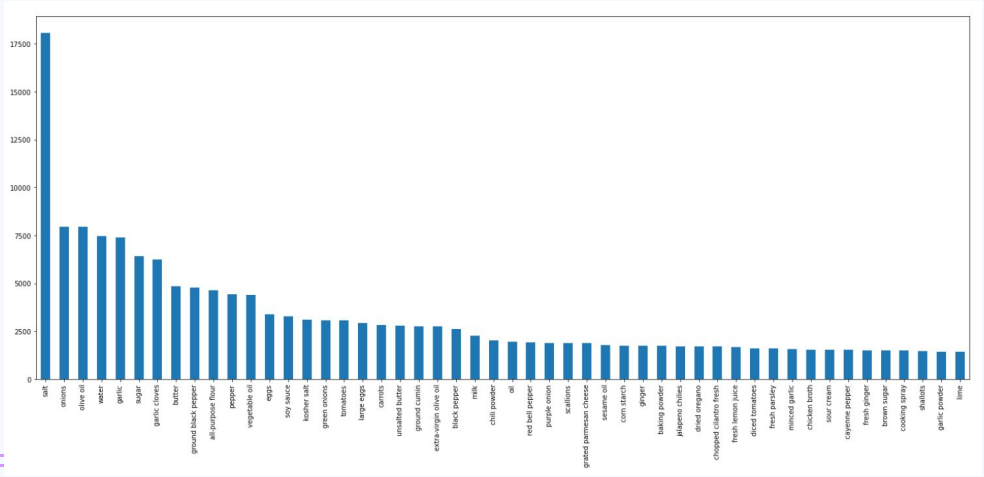
```
#now we plot the occurences of each ingredient in each cuisine, in descending order
cuisine= data_json['cuisine'].unique()
cuisine_dict = dict()
colors = ["b", "g", "r", "c", "m","y", "b", "g", "r", "c","m", "y", "b", "g", "r","c", "m", "y", "b", "g"]
count = 0
for x in cuisine:
    i = []
    for ing_list in data_json[data_json['cuisine']==x]['ingredients']:
        for ing in ing_list:
            i.append(ing)
    cuisine_dict[x] = i
for key in cuisine_dict.keys():
    fig, ax = plt.subplots(figsize=(25,2))
    pd.Series(cuisine_dict[key]).value_counts().head(25).plot.bar(ax=ax, title=key.capitalize(), color = colors[count])
    count+=1
    plt.show()
```



Each Cuisine

Exploratory Analysis

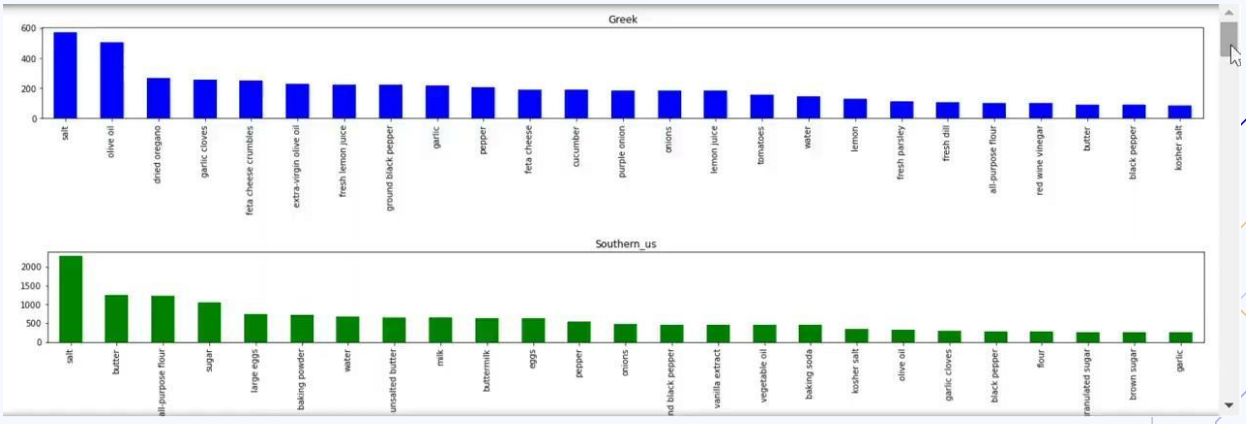
Data Visualisation can be done via catplots.
The following shows the most used ingredients overall.



Data visualisation can be done via catplots. The following shows the most used ingredients overall.

Exploratory Analysis

The following shows the most used ingredients for each cuisine.



The following shows the most used ingredients for each cuisine.

Exploratory Analysis

Our problem statement is for someone with certain dietary restrictions, which cuisine could be recommended to them?

Hence, we can implement some simple commands to filter out the dietary restrictions and display their food indexes and counts.

The more times an ingredient appears, the higher chance of it being consumed.

```
korean_milk_foods = data_df.loc[data_df['Cuisine'] == 'korean']
korean_milk_foods = korean_milk_foods[korean_milk_foods['Ingredients'].str.contains('milk')]
print('The amount of foods containing milk in korean cuisine are '+ str(len(korean_milk_foods)))
```

```
The amount of foods containing milk in korean cuisine are 8
```

```
print('The food indexes to avoid are:')
for x in korean_milk_foods.Index:
    print(str(x))
```

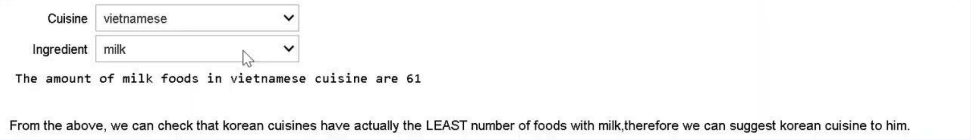
```
The food indexes to avoid are:
8493
9432
13082
13870
28434
32015
36261
38577
```

Exploratory Analysis

We can also implement a user interface where we can choose the cuisine and the ingredient.

E.g. james is allergic to milk

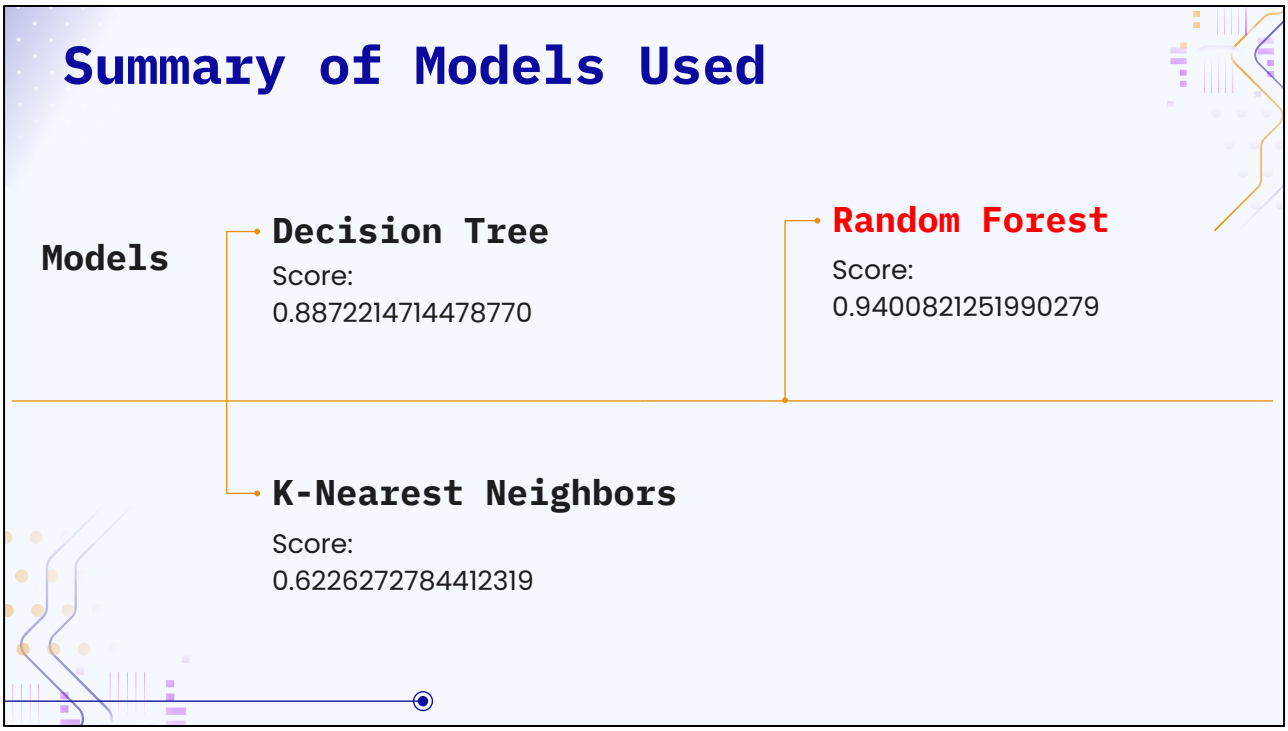
```
from ipywidgets import interact

Ingredient = ['milk', 'sugar', 'egg', 'shrimp', 'peanut']
Cuisine = ['greek', 'indian', 'cajun_creole', 'mexican', 'moroccan', 'filipino', 'brazilian', 'russian', 'french', 'jamaican

@interact(Cuisine=Cuisine, Ingredient=Ingredient)
def count_ingredient_in_cuisine(Cuisine, Ingredient):
    greek_avoid_foods = data_df.loc[data_df['Cuisine'] == Cuisine]
    greek_avoid_foods = greek_avoid_foods[greek_avoid_foods['Ingredients'].str.contains(Ingredient)]
    print(f'The amount of {Ingredient} foods in {Cuisine} cuisine are {len(greek_avoid_foods)}')
```



From the above, we can check that korean cuisines have actually the LEAST number of foods with milk,therefore we can suggest korean cuisine to him.

We can also implement a user interface where we can choose the cuisine and the ingredient. For example, james is allergic to milk. We can check that korean cuisines have actually the least number of foods with milk, therefore we can suggest korean cuisine to him.

Thank you Malvin!

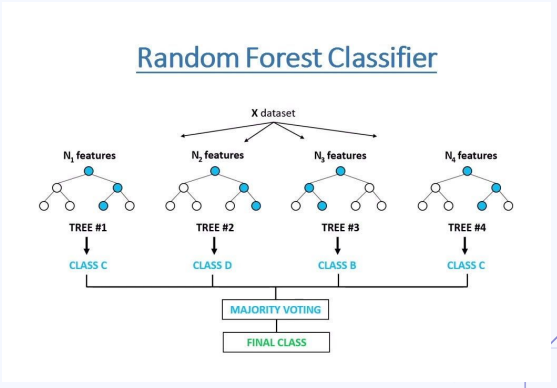


Now we will be moving on to the Machine Learning part. We have decided to use classification for our project and we have tested out on 3 different classification models which includes Decision Tree, K Nearest Neighbour KNN as well as Random Forest. Based on the accuracy score as shown in the slides we decided to use Random Forest out of the 3 as it has the highest accuracy which would mean that it has the highest percentage of correct number of predictions out of the total number of predictions.

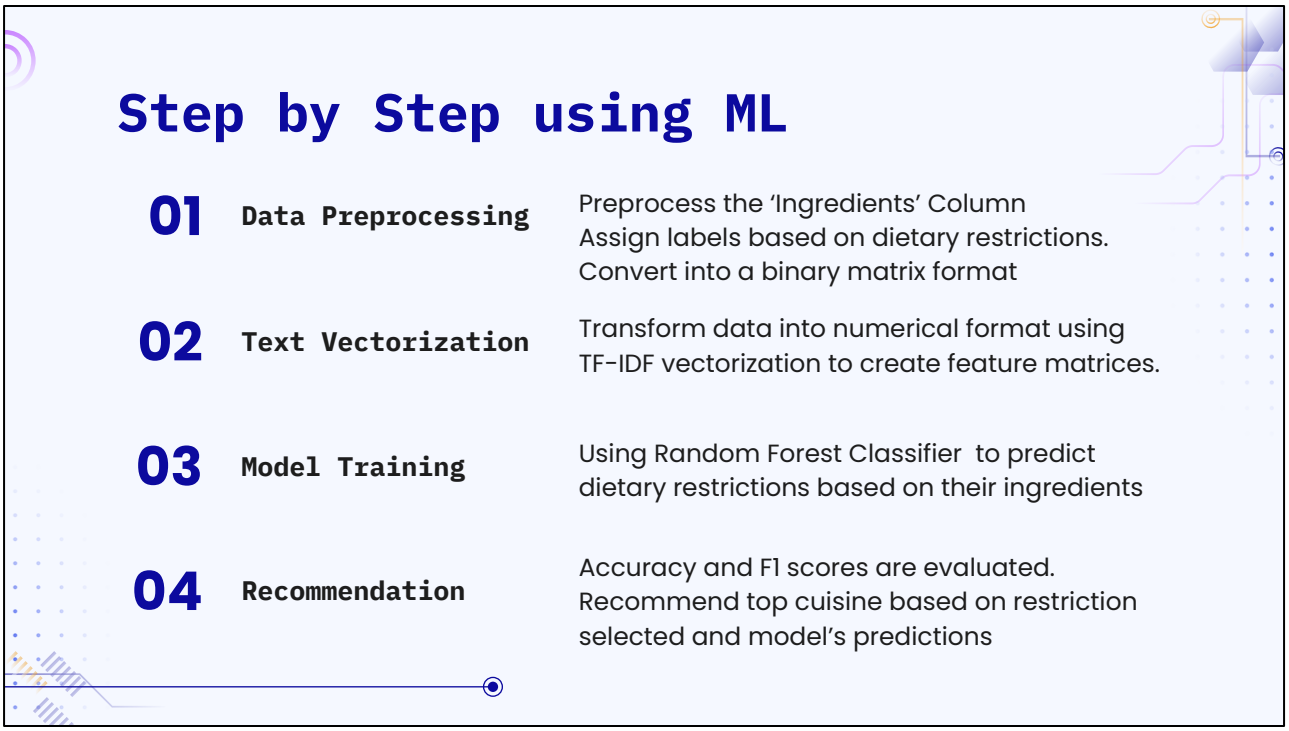
Random Forest for Classification

What is Random Forest Classification?

- Random Forest is an ensemble classifier that consist of a collection of decision trees
- Can handle multi-label classification problem
- It is more efficient and doesn't overfit with more features



Moving on I will first explain what exactly is Random Forest Classification Model as it is something new that we have applied to this project and was not taught in our course content. Random Forest is an ensemble classifier that consists of a collection of decision trees, where predictions are combined to determine the most commonly agreed-upon outcome. It is also suitable for multi label classification which is suitable for solving our problem



Now I will be explaining how we used the Random Forest Classification Model to solve our problem step by step. First, I did some preprocessing on the 'ingredients' column and assign labels to the data based on the dietary restrictions that we have chosen beforehand which are dairy allergies, egg allergy, peanut allergy and seafood allergy. The labels are then converted into a binary matrix format using MultiLabelBinarizer where '1' will represent that the restriction applies and '0' value if it doesn't. Next, the recipe ingredients are transformed into numerical format using TF-IDF vectorization to create feature matrices for training and testing. Next, we split the dataset into both training and test sets and train the Random Forest Classification Model to predict dietary restrictions based on their ingredients in the recipes. Finally after training the model, I calculated and evaluated the accuracy of the model as well as the F1 scores for each dietary restriction. Furthermore, I added a function to recommend the top few cuisines based on user-selected restrictions in a dropdown list and the model's predictions. All these steps were implemented to help solve our objective.

Data Preprocessing & Labelling

```
import re

# Apply preprocess to the 'ingredients' column
def preprocess(ingredients):
    # Converting to lowercase
    ingredients = [x.lower() for x in ingredients]

    # Removing extra white spaces
    ingredients = [re.sub('\s+', ' ', x).strip() for x in ingredients]

    # Removing numbers
    ingredients = [re.sub("\d+", "", x) for x in ingredients]

    # Removing punctuation and special characters
    ingredients = [x.replace("-", " ").replace("$", " ").replace("'", " ").replace("''", " ").replace("%", " ").replace("!", " "
    ingredients = [re.sub(r'[^\w\s]', '', x) for x in ingredients]

    # Removing units of measurement
    units_list = ['g', 'lb', 's', 'n', 'oz']
    ingredients = [remove_units(x, units_list) for x in ingredients]

    # Removing words which are not ingredients
    ingredients = [re.sub(r'crushed|crumbles|ground|minced|powder|chopped|sliced', '', x) for x in ingredients]

    return ' '.join(ingredients)

# Function to remove units from a string
def remove_units(word, units_list):
    s = word.split()
    resw = [word for word in s if word.lower() not in units_list]
    return ' '.join(resw)
```

Code function for data preprocessing to get cleaned data for the machine learning

	id	cuisine	ingredients	labels
0	10259	greek	romaine lettuce black olives grape tomatoes ga...	[Dairy Allergy]
1	25693	southern_us	plain flour pepper salt tomatoes black peppe...	[Dairy Allergy, Egg Allergies]
2	20130	filipino	eggs pepper salt mayonaise cooking oil green c...	[Dairy Allergy, Egg Allergies]
3	22213	indian	water vegetable oil wheat salt	[]
4	13162	indian	black pepper shallots cornflour cayenne pepper...	[Dairy Allergy]
...	...	...	...	...
39769	29109	irish	light brown sugar granulated sugar butter warm...	[Dairy Allergy, Egg Allergies]
39770	11462	italian	kraft zesty italian dressing purple onion broc...	[Dairy Allergy]
39771	2238	irish	eggs citrus fruit raisins sourdough starter fl...	[Dairy Allergy, Egg Allergies]
39772	41882	chinese	boneless chicken skinless thigh garlic steame...	[Egg Allergies, Peanut Allergies]
39773	2362	mexican	green chile jalapeno chilies onions black pep...	[]

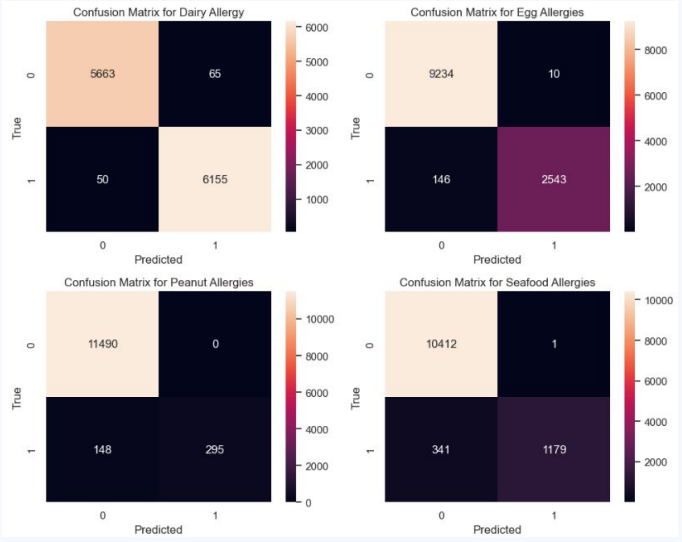
Dataset after preprocessing and labelling based on dietary restrictions

Accuracy and F1 Scores

What is F1 Score?
Fraction of true positive records among the total actual positive records

$$\text{F1 Score: } 2 * \frac{Precision * Recall}{Precision + Recall}$$

```
Accuracy: 0.9400821251990279
F1 Score for Dairy Allergy: 0.9907444668008049
F1 Score for Egg Allergies: 0.9702403662724151
F1 Score for Peanut Allergies: 0.7994579945799457
F1 Score for Seafood Allergies: 0.8733333333333333
```



Now I will be showing the accuracy and F1 scores of our model. The accuracy of our model is around 0.934 and our F1 scores for each of our dietary restrictions are shown in the slides. F1 scores is described as the harmonic mean of the precision and recall of a classification model and it is usually used to get the fraction of true positive records among the total positive records. So our F1 scores for each dietary restriction are relatively high and as for the confusion matrix as shown on the right, it shows that our classification model performs pretty well as it has a high number of true positives and true negatives for all the different dietary restrictions.

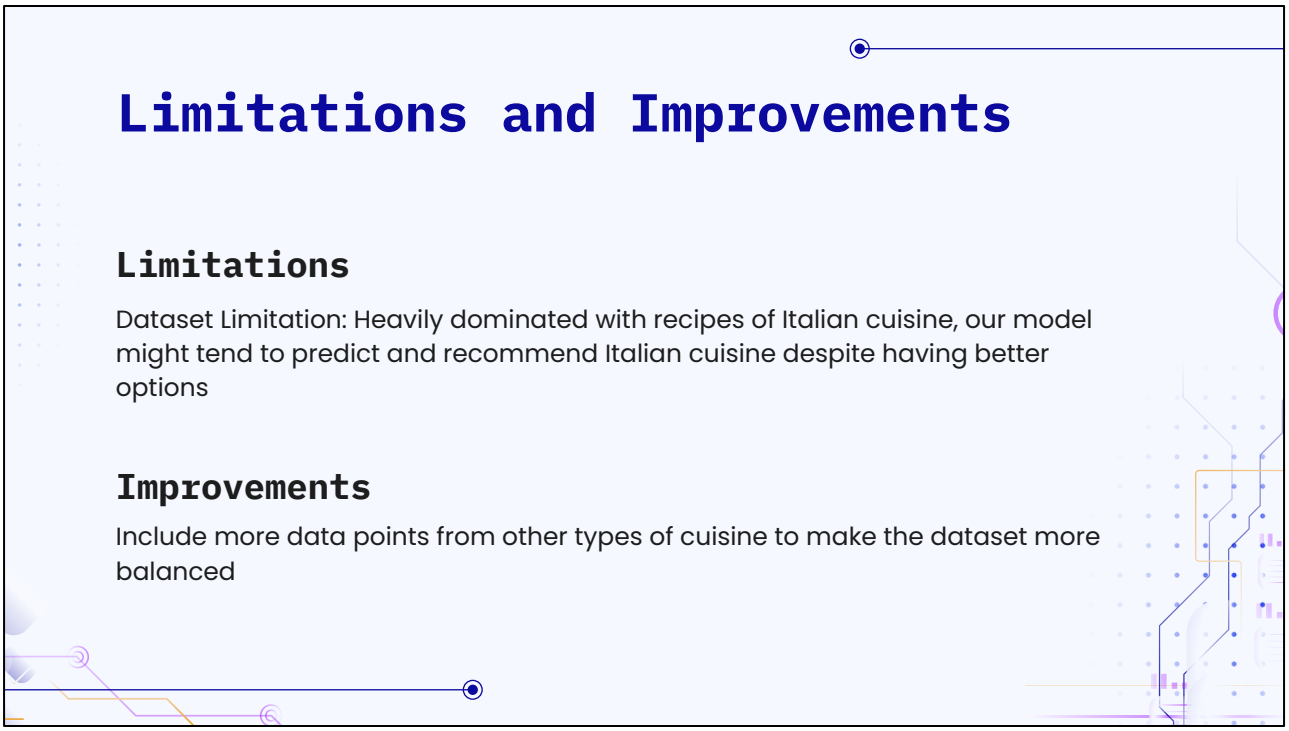
Now I will be passing my time over to lily who will talk about our project outcomes and our member's individual contributions.

Thank you Samantha. Moving on to our outcomes and contributions.

Outcome of our Project

From our classification model, we can conclude that the top recommended cuisines for each dietary restrictions is as follows:

Using the random forest classification, we have concluded that the top recommended cuisines for each dietary restrictions is Indian for Dairy allergies, Mexican for Peanut Allergies and Italian for both egg and seafood allergies. Something interesting is that italian food seems to dominate as the recommended cuisine for allergies.



One limitation we have identified is that our result is dominated by recipes of the Italian cuisine, this may mean our model might tend to predict and recommend Italian cuisines despite having better options. Hence, an improvement we can make is to include more data points from other types of cuisine to make the dataset more balance.

Member's Individual Contribution

Member	Individual Contribution
Malvin	Data Cleaning, Exploratory Analysis, Visualisation
Lily	Used Decision Tree Classification to address the problem statement
Ilakkiyaa	Used K-Nearest Neighbors Classification to address the problem statement
Samantha	Used Random Forest Classification to address the problem statement, Data Preprocessing

For our individual contribution, Malvin did the data cleaning along with exploratory analysis and visualisation while the 3 of us worked on 3 different classification models to address the problem statement. I worked on decision tree classification, Ilakkiyaa worked on the k-nearest neighbours classification while Samantha worked on our chosen model – Random forest classification.

Thank you!