CSCI - 6409 - The Process of Data Science - Fall 2022

</center>

Assignment 1

</center>

Meagan Sinclair
B00737317

Samarth Jariwala
B00899380

1. Business Understanding

Task 1.1

Many online content creation business such as YouTube channels or blogs feature and review various restaurants for their viewers. By featuring restaurants which have received awards, high popularity ratings, or are the most accessible on TripAdvisor they can boost their audience's interest and possibly increase clicks on their pages. Using TripAdvisor data, the content creators can make informed decisions on which restaurants to feature for the best chance at showing future high popularity or highly awarded restaurants.

Task 1.2

Solution 1: The first solution to the problem is to predict which restaurants will have a high, average, or low rating. This will directly help the content creator to decide which restaurant to feature. *Data Requirements:* Input data required for this solution could include any types of information relevant to restaurants' relevance such as current number of reviews, popularity, cuisine type, open hours, tags, price range, and awards as well as the target average rating. *Capacity Requirements:* For this solution to be feasible, the input data used would need to be available for new restaurants in order to assess them and perform the prediction prior to an established rating being published.

Solution 2: The second solution to predict the direct value of a restaurant as an article subject. *Data Requirements:* For this solution, the subject value will need to be quanitified. This could be done by assessing the number of views, clicks, or Google searches for a specific restaurant. *Capacity Requirements:* This solution will require extra datamining to add to the available data.

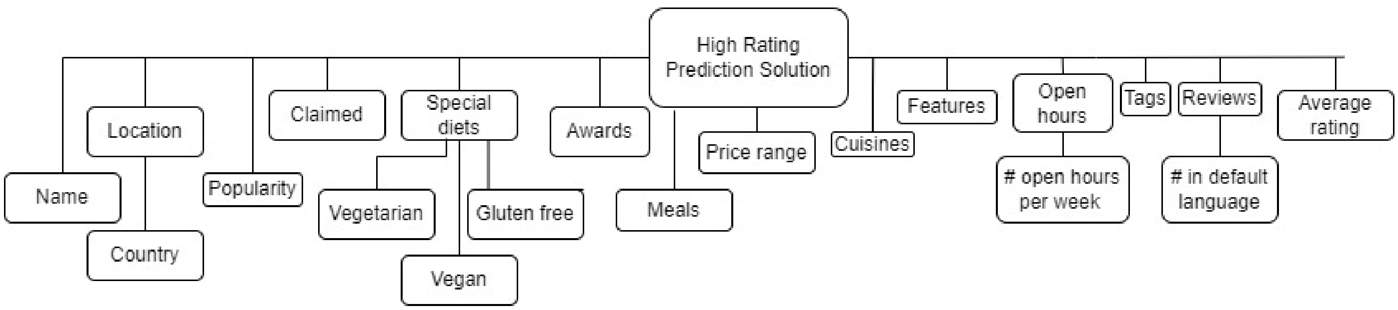
Solution 3: The third solution is to predict whether or not a restaurant will receive an award from TripAdvisor and therefore become a better feature subject. *Data Requirements:* The data required for this would include information about each restuarant labels as either *recieved award* or *no award*. *Capacity Requirements:* The timing of the prediction would depend on when TripAdvisor updates their awards statuses and ensuring that features on restaurants can be completed in time for maximum audience relevance and interest.

Final Solution: The final solution chosen is Solution 1. The rating of the restaurant is the most relevant target to reflect the value of the restaurant as an article subject. The restaurants predicted to have the highest rating will be selected to be featuered. This solution seems to be most liable and also support the problem statement. Further, the required data is currently available for the solution.

Task 1.3

The prediction subject is average rating of the restaurant.

Domain Concepts: Name, Location, Claim Status, Awards, Popularity, Rating, Reviews, Open Hours, Special Diets, Price Range, Features, Meals, Tags, Price Range, Cuisines, Features.



Task 1.4

To create features from Cuisines and Tags, the first category listed will be used as the categorical feature. This is likely the most important data point reflecting the restaurant. The reviews will be transformed into percent of total reviews in the default languauge for each excellent, very good, average, poor, and terrible. This will normalize the data as restaurants will have varying number of total reviews. The restaurant features and meal types offered will be transformed into the number listed available to reflect this. The popularity feature will be quanitfied by taking the number of other restaurants in the area divided by the rank of the particular restaurant. This will provide a number indicitive of the true popularity of the restaurant with consideration to the number of other options in the area. Price range will be transformed into upper price limit and lower price limit. Other features can be used raw. The specific values for food, service, value, and atmosphere ratings will be ignored as the target value is the average of these 4.

Feature Name	Domain Concept	Feature Description	Feature Type	Data Type
Name	Name	Restaurant name	Raw	Textual
Country	Location	The country where the restaurant is located	Raw	Categorical
Popularity Score	Popularity	The value reflecting the popularity	Derived	Numeric
Claimed Status	Claimed	Whether or not the restaurant has claimed the listing	Raw	Binary
Vegetarian	Special Diets	If vegetarian options are offered	Raw	Binary
Vegan	Special Diets	If vegan options are offered	Raw	Binary
Gluten Free	Special Diets	If gluten free options are offered	Raw	Binary
Awards	Awards	Whether the restaurant has received an award	Raw	Binary
Number of meals	Meals	The number of meal types offered (breakfast, lunch, etc)	Derived	Numeric

Upper price range	Price range	The upper price range limit	Raw	Numeric
Lower price range	Price range	The lower price range limit	Raw	Numeric
Top cuisine tag	Cuisines	The first listed cuisine tag	Derived	Categorical
Top tag name	Tags	The first listed tag	Derived	Categorical
Number open hours per week	Open hours	Number of hours the restaurant is open each week	Raw	Numeric
Number of features	Features	Number of features the restaurant displays	Derived	Numeric
Percent excellent reviews	Reviews	Percent of default language reviews	Derived	Numeric
Percent very good reviews	Reviews	Percent of default language reviews	Derived	Numeric
Percent average reviews	Reviews	Percent of default language reviews	Derived	Numeric
Percent poor reviews	Reviews	Percent of default language reviews	Derived	Numeric
Percent terrible reviews	Reviews	Percent of default language reviews	Derived	Numeric

2. Data Exploration

In [1]:

```
import pandas as pd
import warnings
```

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [26]:

```
# load the dataset
#df = pd.read_csv("/content/drive/MyDrive/PDS-6409/A1/tripadvisor_european_restaurants.cs
df = pd.read_csv("/content/drive/MyDrive/tripadvisor_european_restaurants.csv")
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarnin
g: Columns (4) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [27]:

```
# convert data types
df = df.convert_dtypes()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1083397 entries, 0 to 1083396
Data columns (total 42 columns):
 #   Column                              Non-Null Count    Dtype
---  ------                              --------------    -----
 0   restaurant_link                     1083397 non-null  string
 1   restaurant_name                     1083397 non-null  string
 2   original_location                   1083397 non-null  string
 3   country                             1083397 non-null  string
 4   region                              1033074 non-null  string
 5   province                            742765 non-null   string
 6   city                                682712 non-null   string
 7   address                             1083397 non-null  string
 8   latitude                            1067607 non-null  Float64
 9   longitude                           1067607 non-null  Float64
 10  claimed                             1081555 non-null  string
 11  awards                              263133 non-null   string
 12  popularity_detailed                 988409 non-null   string
```

```
 13  popularity_generic                  985605 non-null   string
 14  top_tags                            972763 non-null   string
 15  price_level                         806192 non-null   string
 16  price_range                         304327 non-null   string
 17  meals                               635347 non-null   string
 18  cuisines                            914294 non-null   string
 19  special_diets                       340256 non-null   string
 20  features                            317407 non-null   string
 21  vegetarian_friendly                 1083397 non-null  string
 22  vegan_options                       1083397 non-null  string
 23  gluten_free                         1083397 non-null  string
 24  original_open_hours                 593832 non-null   string
 25  open_days_per_week                  593832 non-null   Int64
 26  open_hours_per_week                 593832 non-null   Float64
 27  working_shifts_per_week             593832 non-null   Int64
 28  avg_rating                          986761 non-null   Float64
 29  total_reviews_count                 1031162 non-null  Int64
 30  default_language                    988204 non-null   string
 31  reviews_count_in_default_language   988204 non-null   Int64
 32  excellent                           988204 non-null   Int64
 33  very_good                           988204 non-null   Int64
 34  average                             988204 non-null   Int64
 35  poor                                988204 non-null   Int64
 36  terrible                            988204 non-null   Int64
 37  food                                599325 non-null   Float64
 38  service                             604287 non-null   Float64
 39  value                               602692 non-null   Float64
 40  atmosphere                          261785 non-null   Float64
 41  keywords                            99198 non-null    string
dtypes: Float64(8), Int64(9), string(25)
memory usage: 364.7 MB
```

2.1 Data Quality Report - Continous Features Report

In [ ]:

```python
# code source: Tutorial

def build_continuous_features_report(data_df):

    stats = {
        "Count": len,
        "Miss %": lambda df: df.isna().sum() / len(df) * 100,
        "Card.": lambda df: df.nunique(),
        "Min": lambda df: df.min(),
        "1st Qrt.": lambda df: df.quantile(0.25),
        "Mean": lambda df: df.mean(),
        "Median": lambda df: df.median(),
        "3rd Qrt": lambda df: df.quantile(0.75),
        "Max": lambda df: df.max(),
        "Std. Dev.": lambda df: df.std(),
    }

    contin_feat_names = data_df.select_dtypes("number").columns
    continuous_data_df = data_df[contin_feat_names]

    report_df = pd.DataFrame(index=contin_feat_names, columns=stats.keys())

    for stat_name, fn in stats.items():
        # NOTE: ignore warnings for empty features
        with warnings.catch_warnings():
            warnings.simplefilter("ignore", category=RuntimeWarning)
            report_df[stat_name] = fn(continuous_data_df)

    return report_df
```

In [ ]:
```
build_continuous_features_report(df)
```

Out[ ]:

	Count	Miss %	Card.	Min	1st Qrt.	Mean	Median	3rd Qı
latitude	1083397	1.457453	857920	27.640310	41.90986	46.567182	46.58510	51.40536
longitude	1083397	1.457453	969586	-71.218094	-0.802732	5.838040	5.64653	12.23767
open_days_per_week	1083397	45.187960	7	1.000000	6.0	6.327081	7.00000	7.
open_hours_per_week	1083397	45.187960	3105	0.000000	39.0	62.023282	58.50000	81.
working_shifts_per_week	1083397	45.187960	15	1.000000	6.0	7.630754	7.00000	7.
avg_rating	1083397	8.919722	9	1.000000	3.5	4.035943	4.00000	4.
total_reviews_count	1083397	4.821409	3363	0.000000	6.0	102.888989	24.00000	93.
reviews_count_in_default_language	1083397	8.786530	2415	1.000000	2.0	44.563415	7.00000	26.
excellent	1083397	8.786530	1708	0.000000	1.0	24.653440	3.00000	13.
very_good	1083397	8.786530	832	0.000000	0.0	10.490516	2.00000	6.
average	1083397	8.786530	458	0.000000	0.0	4.109302	1.00000	2.
poor	1083397	8.786530	305	0.000000	0.0	2.355306	0.00000	1.
terrible	1083397	8.786530	353	0.000000	0.0	2.954850	0.00000	2.
food	1083397	44.680943	9	1.000000	4.0	4.104179	4.00000	4.
service	1083397	44.222940	9	1.000000	4.0	4.067245	4.00000	4.
value	1083397	44.370162	9	1.000000	3.5	3.982897	4.00000	4.
atmosphere	1083397	75.836651	9	1.000000	3.5	3.933682	4.00000	4.

2.1 Data Quality Report - Categorical Features Report

In [ ]:
```python
# code source: Tutorial

def build_categorical_features_report(data_df):

    """Build tabular report for categorical features"""

    def _mode(df):
        return df.apply(lambda ft: ft.mode().to_list())

    def _mode_freq(df):
        return df.apply(lambda ft: ft.value_counts()[ft.mode()].sum())

    def _second_mode(df):
        return df.apply(lambda ft: ft[~ft.isin(ft.mode())].mode().to_list())

    def _second_mode_freq(df):
        return df.apply(
            lambda ft: ft[~ft.isin(ft.mode())]
            .value_counts()[ft[~ft.isin(ft.mode())].mode()]
            .sum()
        )

    stats = {
        "Count": len,
        "Miss %": lambda df: df.isna().sum() / len(df) * 100,
        "Card.": lambda df: df.nunique(),
        "Mode": _mode,
```

```python
        "Mode Freq": _mode_freq,
        "Mode %": lambda df: _mode_freq(df) / len(df) * 100,
        "2nd Mode": _second_mode,
        "2nd Mode Freq": _second_mode_freq,
        "2nd Mode %": lambda df: _second_mode_freq(df) / len(df) * 100,
    }

    cat_feat_names = data_df.select_dtypes(exclude="number").columns
    continuous_data_df = data_df[cat_feat_names]

    report_df = pd.DataFrame(index=cat_feat_names, columns=stats.keys())

    for stat_name, fn in stats.items():
        # NOTE: ignore warnings for empty features
        with warnings.catch_warnings():
            warnings.simplefilter("ignore", category=RuntimeWarning)
            report_df[stat_name] = fn(continuous_data_df)

    return report_df
```

In [ ]:

```python
build_categorical_features_report(df)
```

Out[ ]:

	Count	Miss %	Card.	Mode	Mode Freq	Mode %	2nd Mode	2nd Mode Freq	2ı Mode
restaurant_link	1083397	0.000000	1083397	[g10001637-d10002227, g10001637-d14975787, g10...	1083397	100.000000	[]	0	0.0000
restaurant_name	1083397	0.000000	840914	[Subway]	4881	0.450527	[McDonald's]	4458	0.4114
original_location	1083397	0.000000	65997	[["Europe", "United Kingdom (UK)", "England", ...	22942	2.117599	[["Europe", "France", "Ile-de-France", "Paris"]]	18129	1.6733
country	1083397	0.000000	24	[Italy]	224763	20.746135	[Spain]	157479	14.5356
region	1083397	4.644927	250	[Lombardy]	33097	3.054928	[Ile-de-France]	31271	2.8863
province	1083397	31.441106	1333	[Province of Barcelona]	18952	1.749313	[Province of Malaga]	10056	0.9281
city	1083397	36.984134	43495	[Paris]	18129	1.673348	[Rome]	12603	1.1632
address	1083397	0.000000	1034685	[Greece]	92	0.008492	[Tsilivi (Planos) 29100 Greece]	29	0.0026
claimed	1083397	0.170021	2	[Unclaimed]	607159	56.042153	[Claimed]	474396	43.7878
awards	1083397	75.712227	917	[Travellers' Choice, Certificate of Excellence...	20868	1.926164	[Certificate of Excellence 2017]	16392	1.5130
popularity_detailed	1083397	8.767608	981409	[#7616 of 8661 Restaurants in Barcelona]	119	0.010984	[#8393 of 10193 Restaurants in Madrid]	99	0.0091

	Count	Miss %	Card.	Mode	Mode Freq	Mode %	2nd Mode	2nd Mode Freq	2 Mode
popularity_generic	1083397	9.026423	981940	[#1 of 1 places to eat in Agios Ioannis]	6	0.000554	[#1 of 1 places to eat in Clifton, #1 of 1 pla...	10	0.0009
top_tags	1083397	10.211769	39962	[Mid-range, French]	20211	1.865521	[Mid-range]	19422	1.7926
price_level	1083397	25.586650	3	[€€-€€€]	537918	49.651051	[€]	240205	22.1714
price_range	1083397	71.909928	7298	[€10-€30]	5937	0.547999	[€5-€15]	5810	0.5362
meals	1083397	41.356031	745	[Lunch, Dinner]	196123	18.102598	[Dinner]	67459	6.2266
cuisines	1083397	15.608590	97741	[Italian]	53243	4.914450	[French]	39103	3.6092
special_diets	1083397	68.593600	68	[Vegetarian Friendly]	156652	14.459335	[Vegetarian Friendly, Vegan Options, Gluten Fr...	71379	6.5884
features	1083397	70.702614	56453	[Reservations]	36514	3.370325	[Reservations, Seating, Table Service]	15193	1.4023
vegetarian_friendly	1083397	0.000000	2	[N]	759380	70.092496	[Y]	324017	29.9075
vegan_options	1083397	0.000000	2	[N]	946800	87.391787	[Y]	136597	12.6082
gluten_free	1083397	0.000000	2	[N]	959900	88.600947	[Y]	123497	11.3990
original_open_hours	1083397	45.187960	237890	[{"Mon": ["00:00-23:59"], "Tue": ["00:00-23:59...	7674	0.708328	[{"Mon": ["11:00-23:00"], "Tue": ["11:00-23:00...	5303	0.4894
default_language	1083397	8.786530	2	[English]	689754	63.665858	[All languages]	298450	27.5476
keywords	1083397	90.843800	99001	[steak, onion loaf, lettuce wedge, chateaubria...	7	0.000646	[curry, poppadoms, rice, lamb, best indian, cu...	12	0.0011

Process raw data into our desired features:

In [282...

```
# dataframe for our selected features
final_features = pd.DataFrame()

# add raw features from data
final_features.insert(0, "name", df["restaurant_name"])
final_features.insert(1, "country", df["country"])
final_features.insert(2, "claimed", df["claimed"])
final_features.insert(3, "veg", df["vegetarian_friendly"])
final_features.insert(4, "vegan", df["vegan_options"])
final_features.insert(5, "gf", df["gluten_free"])
final_features.insert(6, "open_hours", df["open_hours_per_week"])
```

```
#change awards into either Y or N
k = df["awards"].fillna('N')
k[k != 'N'] = 'Y'
final_features.insert(7, "awards", k)
```

In [260…

```
# derive aggregate features
pop_score = []
top_tag = []
top_cuisine = []
upper_price = []
lower_price = []
num_features = []
num_meals = []
p_excellent = []
p_vgood = []
p_average = []
p_poor = []
p_terrible = []

# for each restaurant, get the corresponding features
for index, row in df.iterrows():

  # get the popularity score
  try:
    nums = row["popularity_detailed"][1:].split(" ")
    #number in area divided by the rank
    pop_score.append(float(nums[2])/float(nums[0]))
  except:
    # missing or invalid value, replace with 0
    pop_score.append(0)

  # get the top top tag
  try:
    tags = row["top_tags"].split(",")
    top_tag.append( tags[0])
  except:
     # missing or invalid value, replace with a blank
     top_tag.append('')

  # get the top cuisine tag
  try:
    tc = row["cuisines"].split(",")
    top_cuisine.append(tc[0])
  except:
     # missing or invalid value, replace with a blank
     top_cuisine.append('')

  # get the upper and lower limit price values
  try:
    nums = row["price_range"].split("€")
    upper_price.append(int(nums[2].replace(',', '')))
    lower_price.append(int(nums[1][:-1].replace(',', '')))
  except:
    # missing or invalid value, replace with 0s
    upper_price.append(0)
    lower_price.append(0)


  # add number of features and meals offered
  try:
    num_features.append(len(row["features"].split(",")))
  except:
    # missing or invalid value, replace with 0
    num_features.append(0)
```

```python
    try:
      num_meals.append(len(row["meals"].split(",")))
    except:
      # missing or invalid value, replace with 0
      num_meals.append(0)

    # calculate proportions of reviews which are excellent, vgood, average, poor, terrible
    try:
      num_reviews = float(row["reviews_count_in_default_language"])
      p_excellent.append(float(row["excellent"])/num_reviews)
      p_vgood.append(float(row["very_good"])/num_reviews)
      p_average.append(float(row["average"])/num_reviews)
      p_poor.append(float(row["poor"])/num_reviews)
      p_terrible.append(float(row["terrible"])/num_reviews)
    except:
      # missing or invalid value, replace with 0s
      p_excellent.append(0)
      p_vgood.append(0)
      p_average.append(0)
      p_poor.append(0)
      p_terrible.append(0)
```

In [261…

```python
# add to data frame
final_features.insert(8, "pop_score", pop_score)
final_features.insert(9, "top_tag", top_tag)
final_features.insert(10, "top_cuisine", top_cuisine)
final_features.insert(11, "upper_price", upper_price)
final_features.insert(12, "lower_price", lower_price)
final_features.insert(13, "num_features", num_features)
final_features.insert(14, "num_meals",num_meals)
final_features.insert(15, "p_excellent", p_excellent)
final_features.insert(16, "p_vgood", p_vgood)
final_features.insert(17, "p_average",p_average)
final_features.insert(18, "p_poor",p_poor)
final_features.insert(19, "p_terrible",p_terrible)

# add target to final column
final_features.insert(20, "ave_rating", df["avg_rating"])
```

In [262…

```python
del pop_score
del top_tag
del top_cuisine
del upper_price
del lower_price
del num_features
del num_meals
del p_excellent
del p_vgood
del p_average
del p_poor
del p_terrible
```

In [263…

```python
final_features = final_features.convert_dtypes()
final_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1083397 entries, 0 to 1083396
Data columns (total 21 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
```

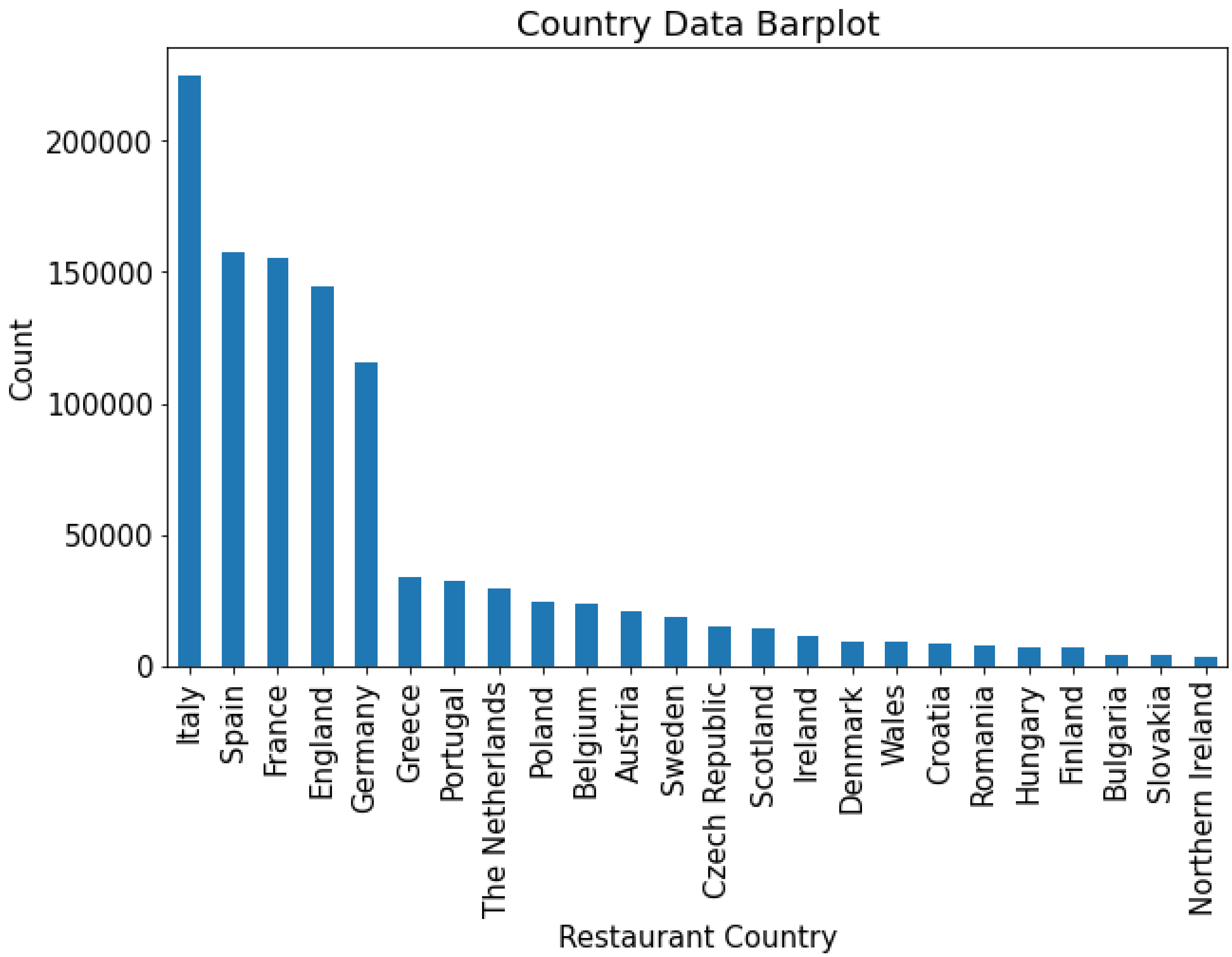
```
 0   name          1083397 non-null  string
 1   country       1083397 non-null  string
 2   claimed       1081555 non-null  string
 3   veg           1083397 non-null  string
 4   vegan         1083397 non-null  string
 5   gf            1083397 non-null  string
 6   open_hours    593832 non-null   Float64
 7   awards        1083397 non-null  string
 8   pop_score     1083397 non-null  Float64
 9   top_tag       1083397 non-null  string
 10  top_cuisine   1083397 non-null  string
 11  upper_price   1083397 non-null  Int64
 12  lower_price   1083397 non-null  Int64
 13  num_features  1083397 non-null  Int64
 14  num_meals     1083397 non-null  Int64
 15  p_excellent   1083397 non-null  Float64
 16  p_vgood       1083397 non-null  Float64
 17  p_average     1083397 non-null  Float64
 18  p_poor        1083397 non-null  Float64
 19  p_terrible    1083397 non-null  Float64
 20  ave_rating    986761 non-null   Float64
dtypes: Float64(8), Int64(4), string(9)
memory usage: 186.0 MB
```

In [95]:

```python
from matplotlib import pyplot as plt

# Set the figure size - handy for larger output
plt.rcParams["figure.figsize"] = [10, 6]
plt.rcParams["font.size"] = 15
```

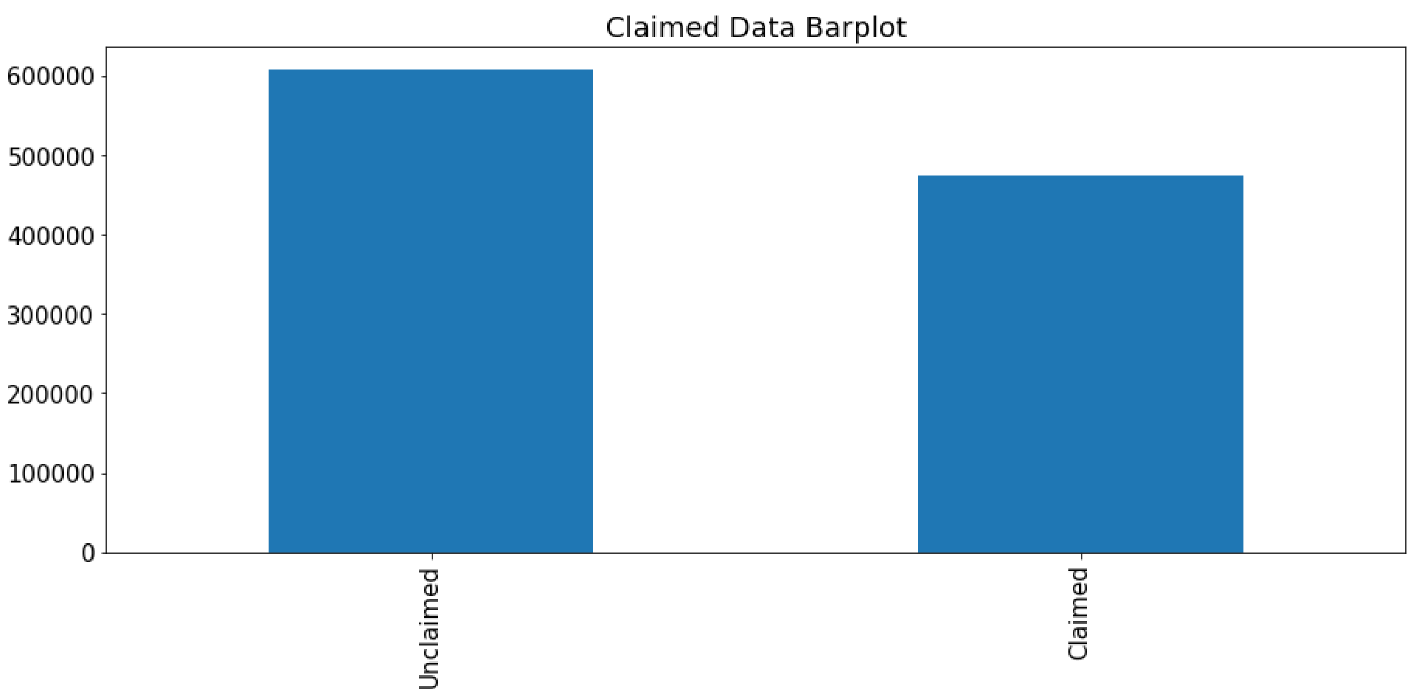
In [101…

```python
final_features['country'].value_counts().plot.bar()
plt.xlabel("Restaurant Country")
plt.ylabel("Count")
plt.title("Country Data Barplot")
plt.show()
```



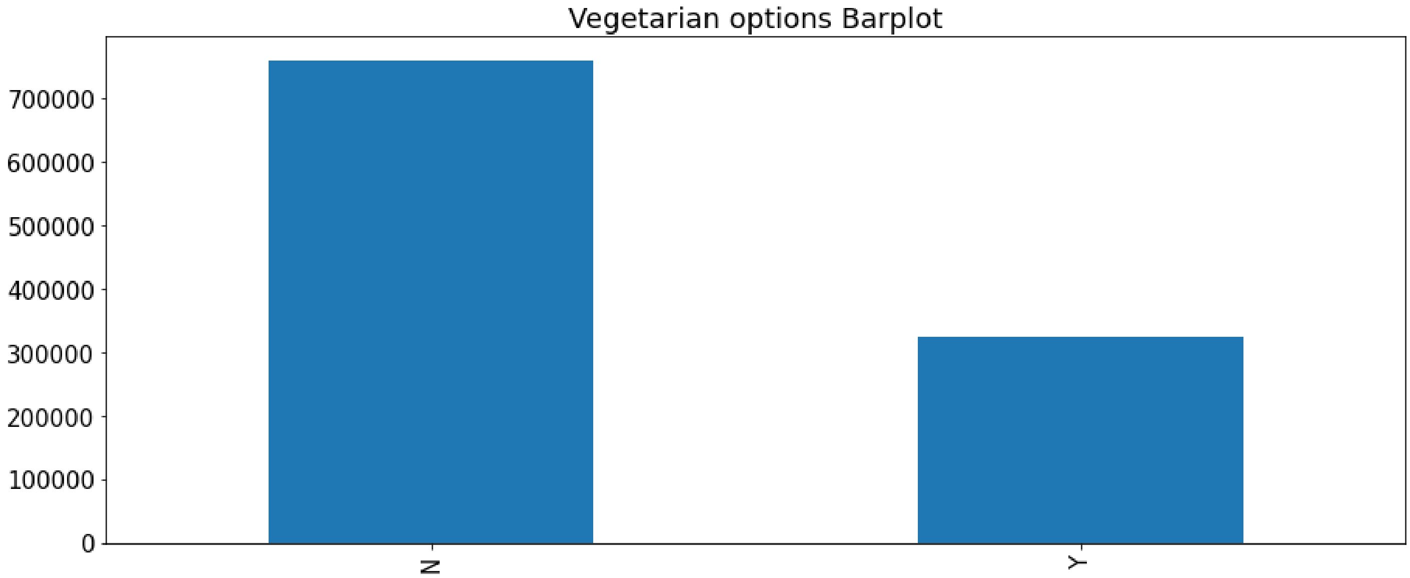
In [161…

```
final_features['claimed'].value_counts().plot.bar()
plt.title("Claimed Data Barplot")
plt.show()
```



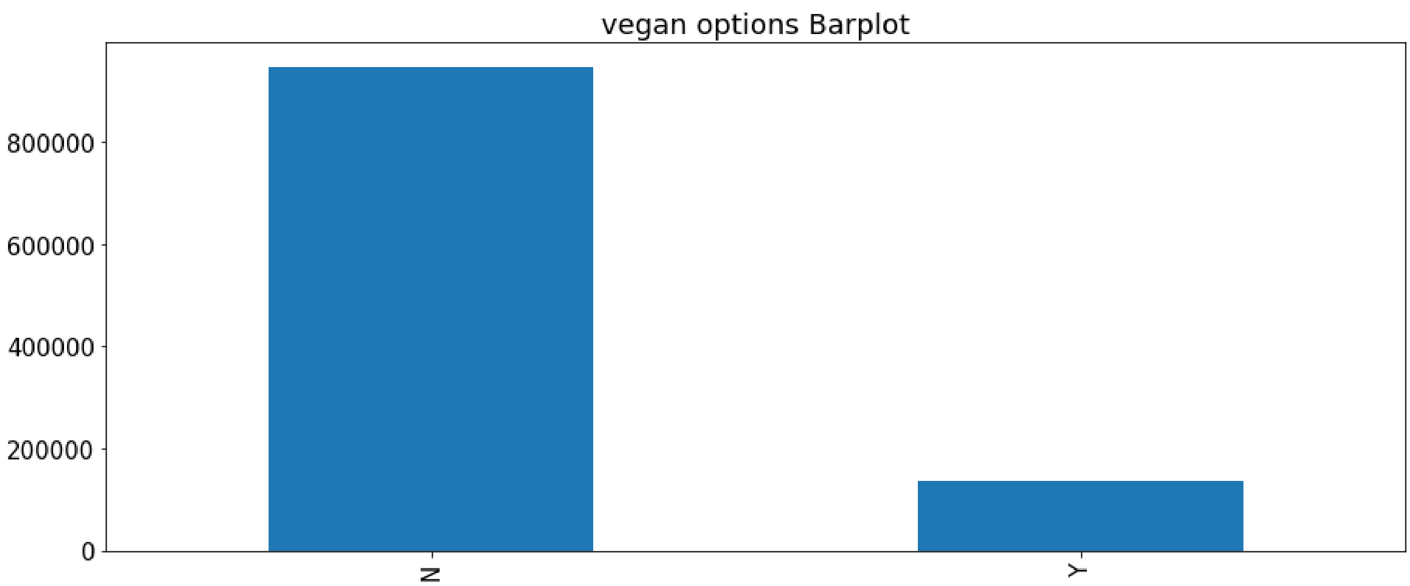
In [162…

```
final_features['veg'].value_counts().plot.bar()
plt.title("Vegetarian options Barplot")
plt.show()
```



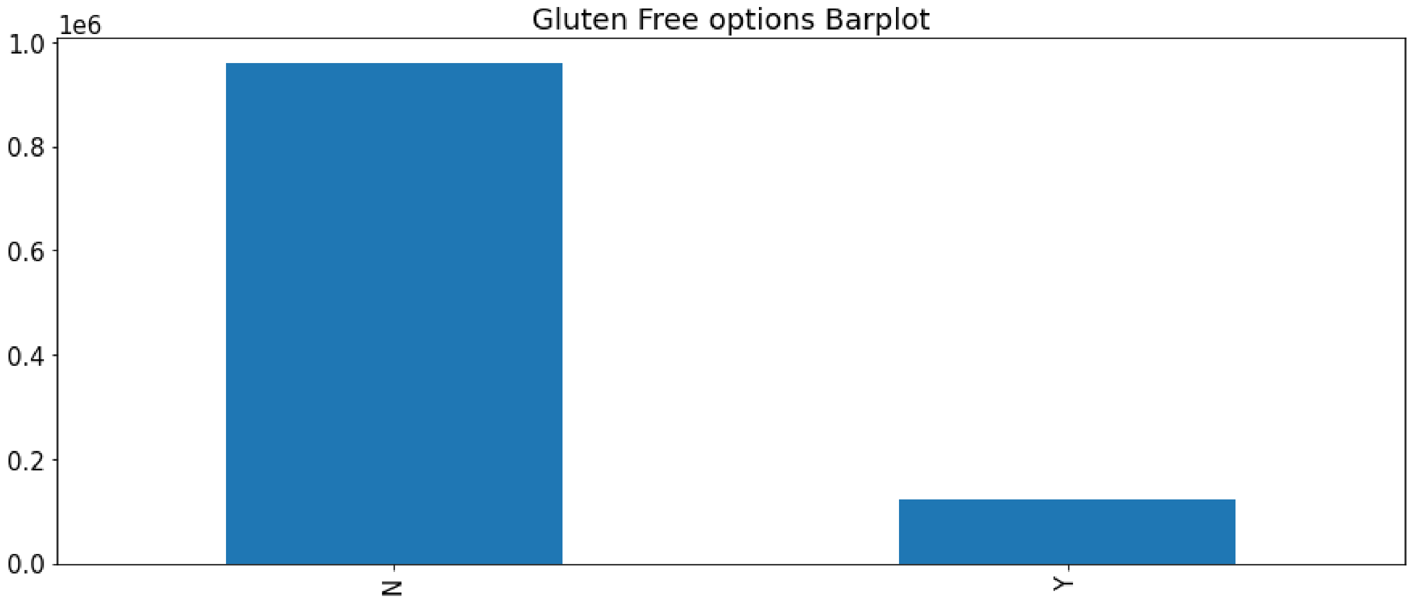
In [163…

```
final_features['vegan'].value_counts().plot.bar()
plt.title("vegan options Barplot")
plt.show()
```



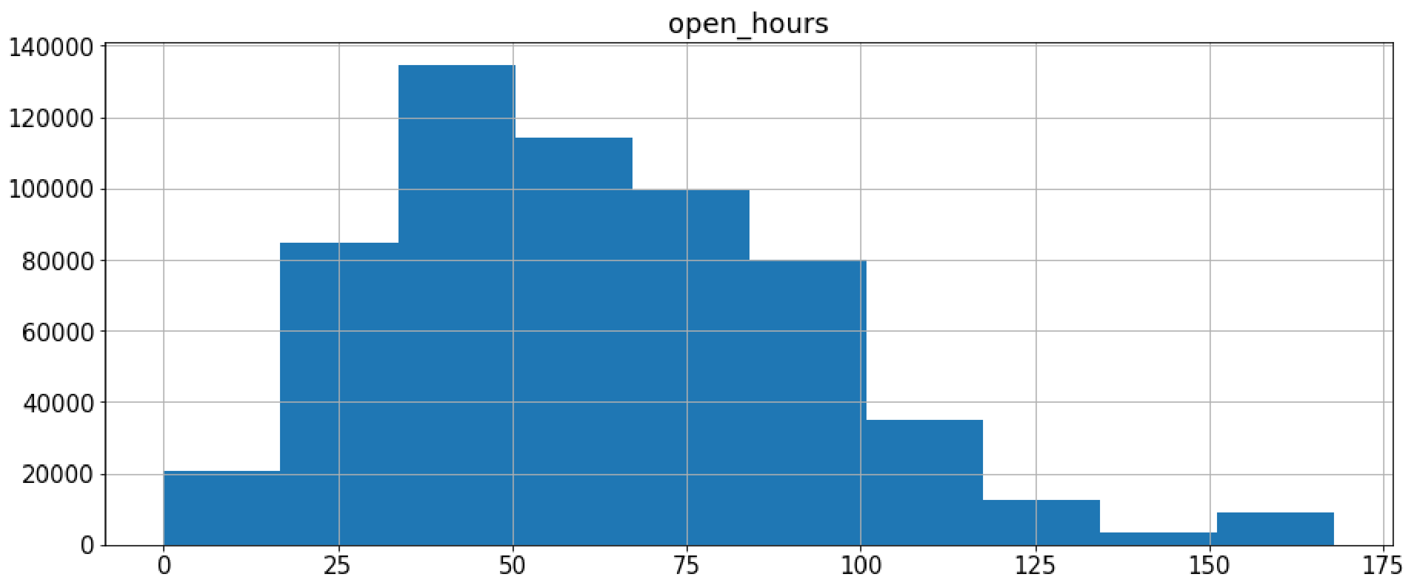
In [164…

```
final_features['gf'].value_counts().plot.bar()
plt.title("Gluten Free options Barplot")
plt.show()
```



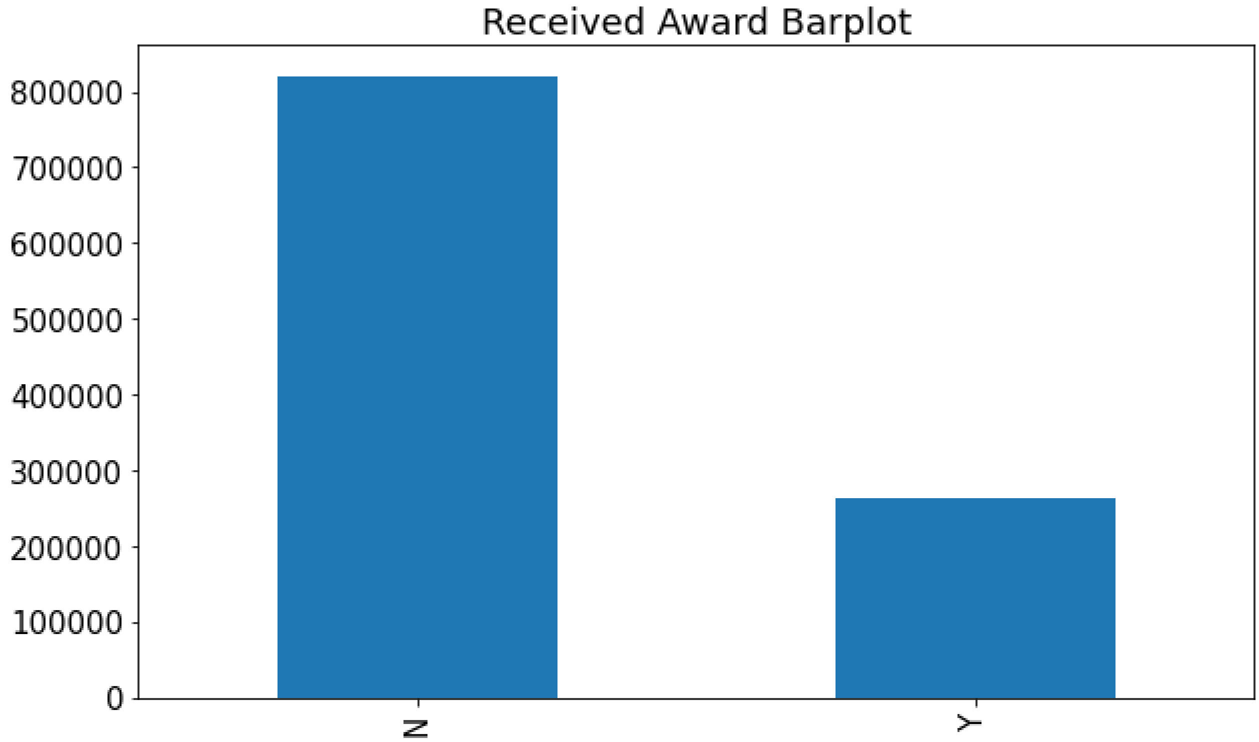
In [283…

```
final_features.hist(column='open_hours')
plt.show()
```



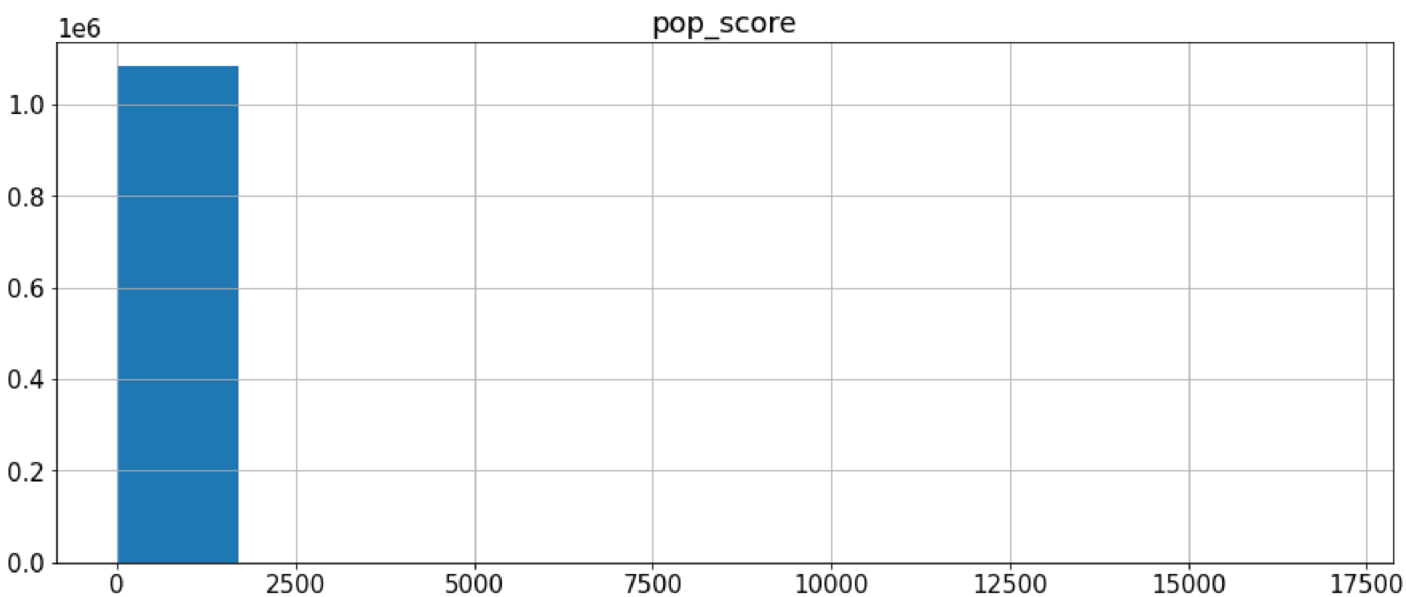
In [110…

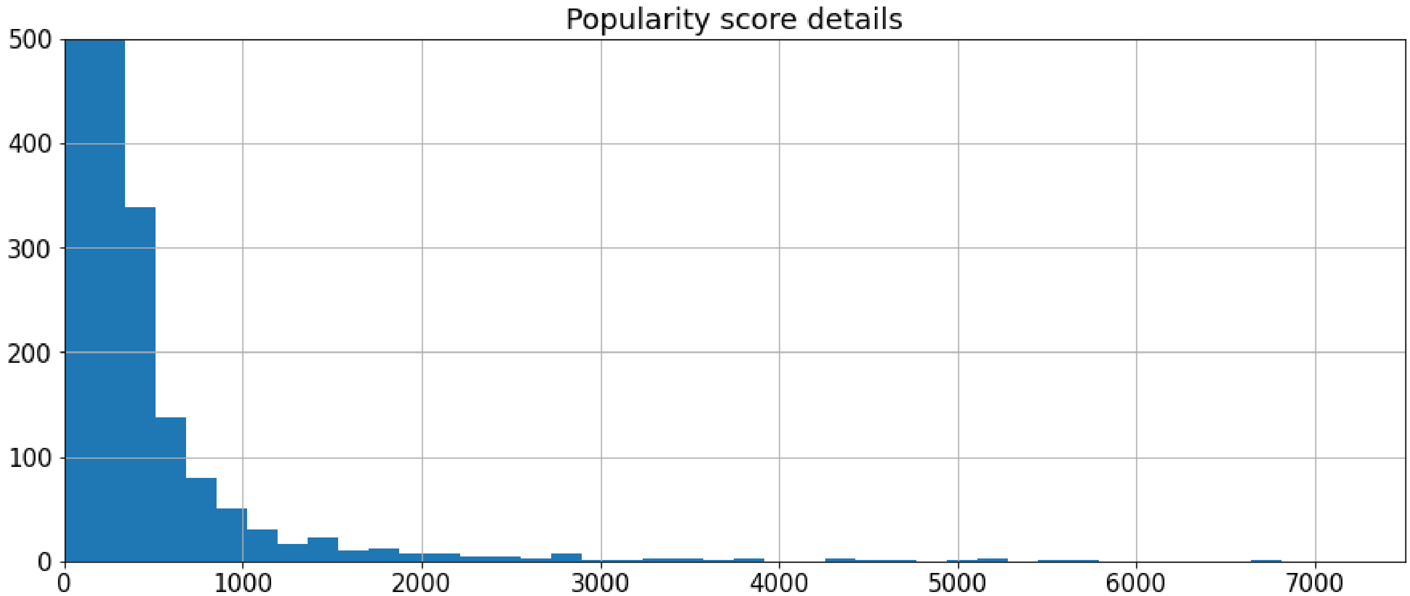
```
final_features['awards'].value_counts().plot.bar()
plt.title("Received Award Barplot")
plt.show()
```



In [264…

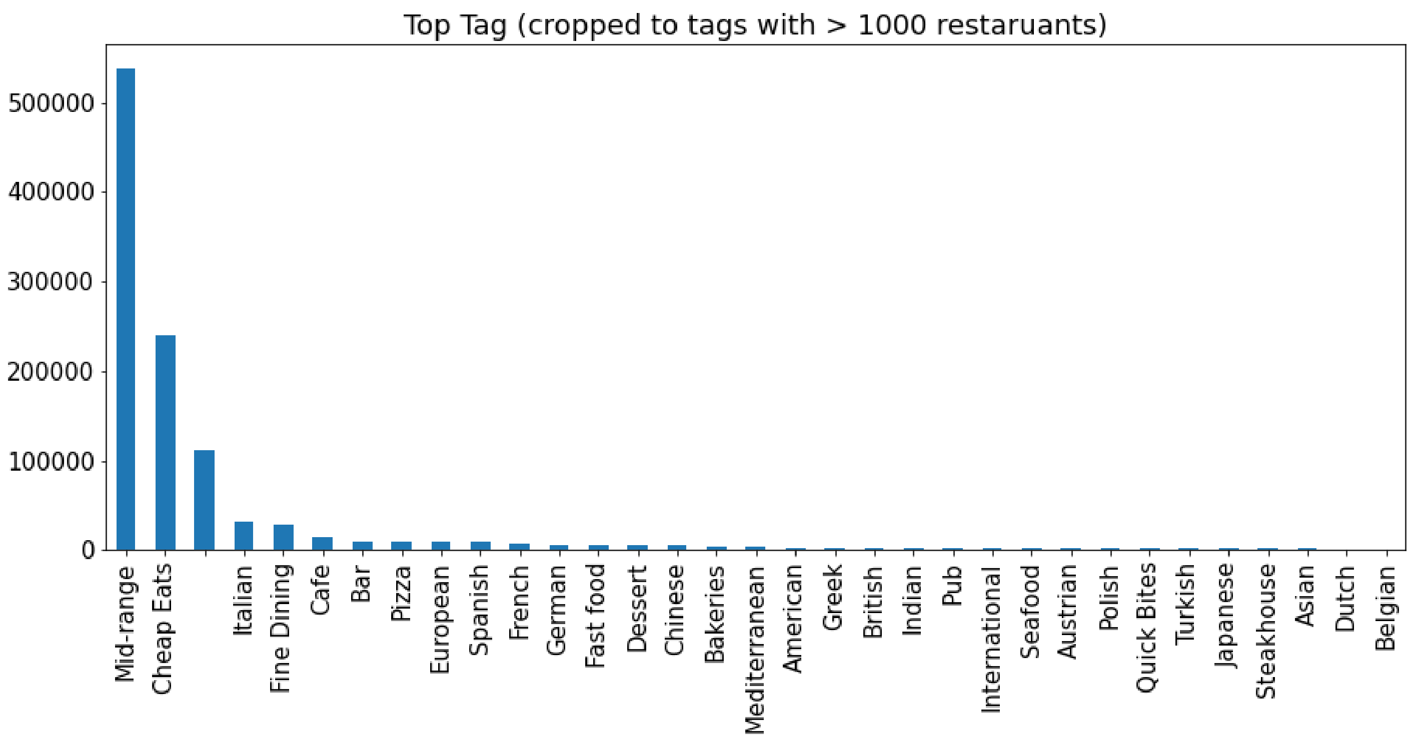
```
final_features.hist(column='pop_score')
final_features.hist(column='pop_score', bins=100)
plt.ylim([0,500])
plt.xlim([0,7500])
plt.title("Popularity score details")
plt.show()
```





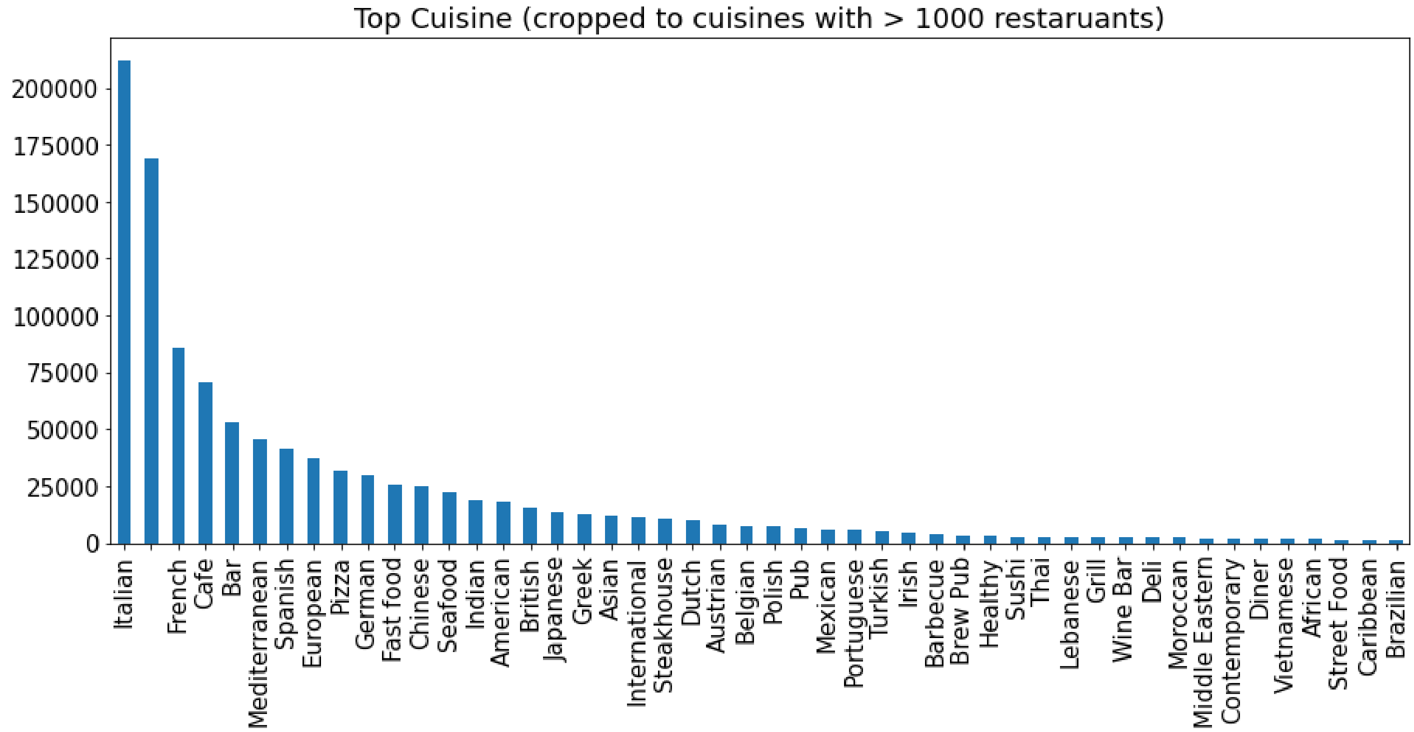
In [128…

```
plt.rcParams["figure.figsize"] = [15, 6]
vcounts = final_features['top_tag'].value_counts()

vcounts[vcounts > 1000].plot.bar()
plt.title("Top Tag (cropped to tags with > 1000 restaruants)")
plt.show()
```



In [125…

```
plt.rcParams["figure.figsize"] = [15, 6]
vcounts = final_features['top_cuisine'].value_counts()

vcounts[vcounts > 1000].plot.bar()
plt.title("Top Cuisine (cropped to cuisines with > 1000 restaruants)")
plt.show()
```

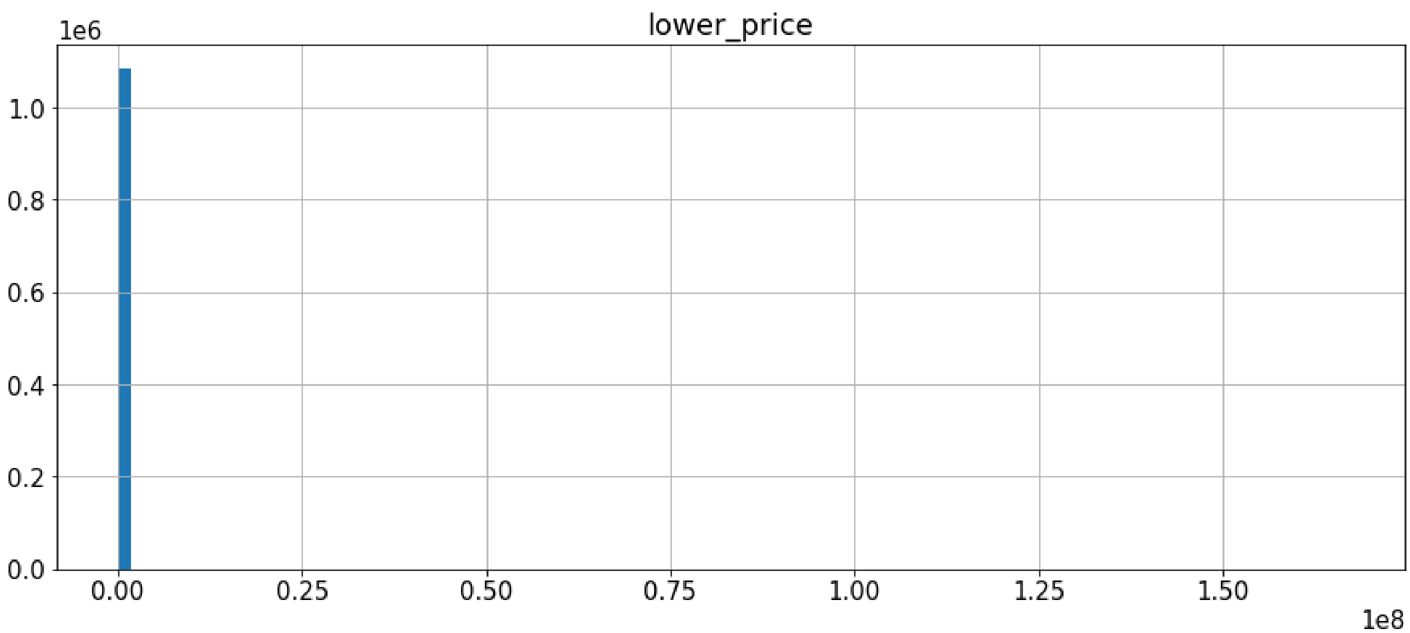


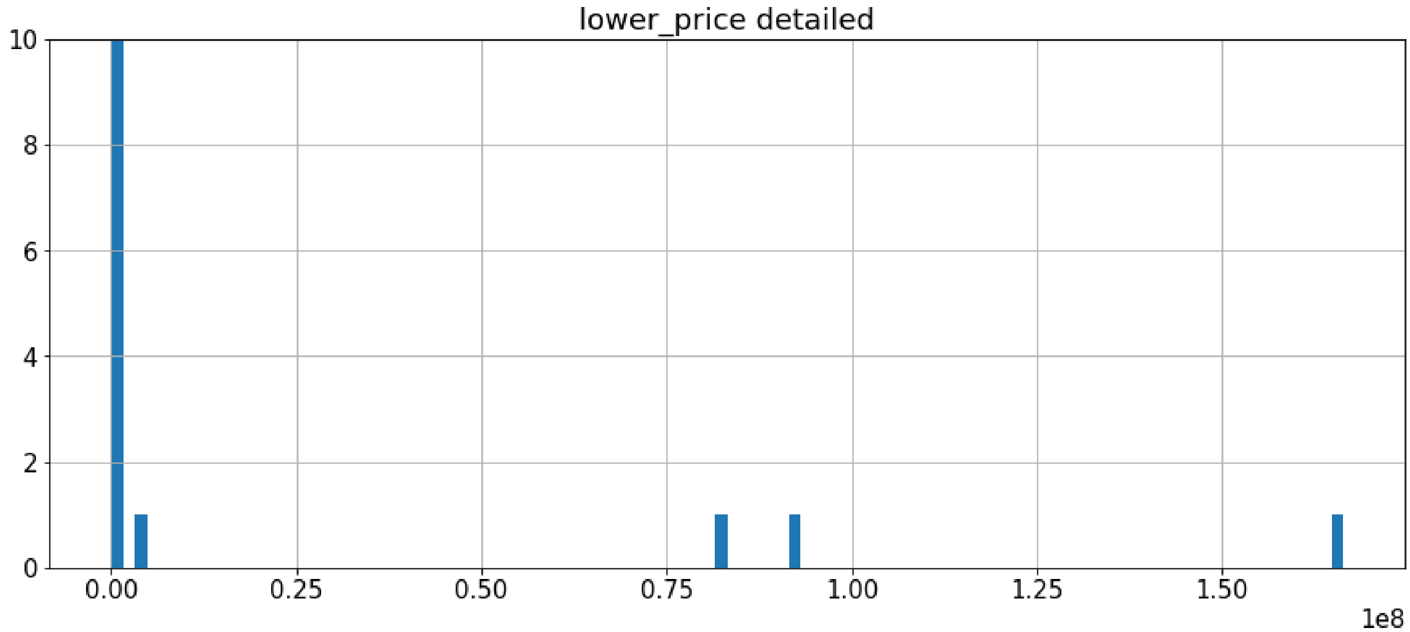
In [224…

```
final_features.hist(column='upper_price', bins=100)
final_features.hist(column='upper_price', bins=100)
plt.title("upper_price detailed")
plt.ylim([0, 10])
plt.show()
```

In [225…

```
final_features.hist(column='lower_price', bins=100)
final_features.hist(column='lower_price', bins=100)
plt.title("lower_price detailed")
plt.ylim([0, 10])
plt.show()
```



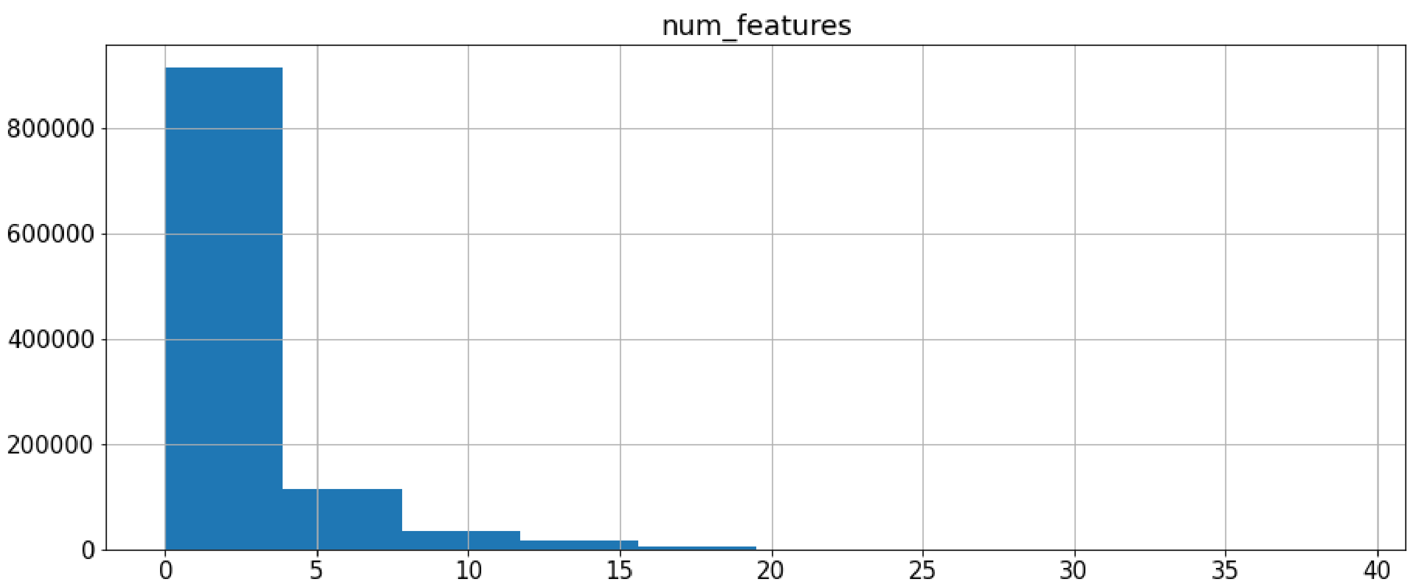


In [201…

```
final_features.hist(column='num_features')
plt.show()
```

Out[201…

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8b5f244310>]],
      dtype=object)
```

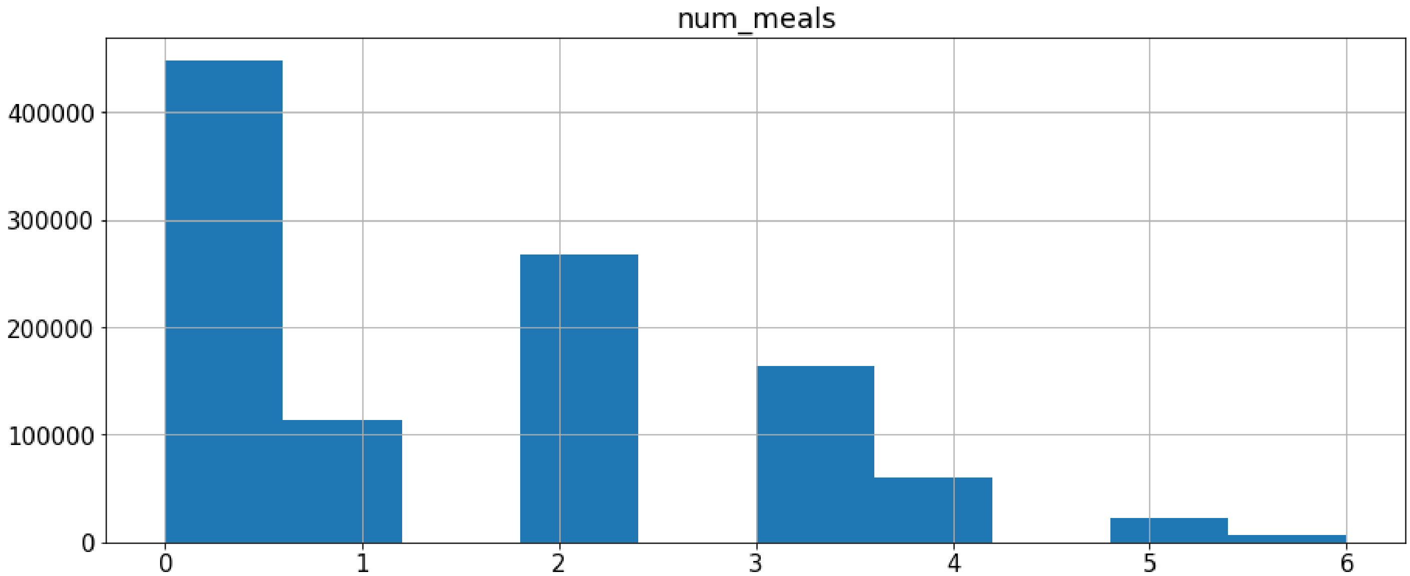


In [202…

```
final_features.hist(column='num_meals')
plt.show()
```

Out[202…

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8b5f4e9750>]],
      dtype=object)
```

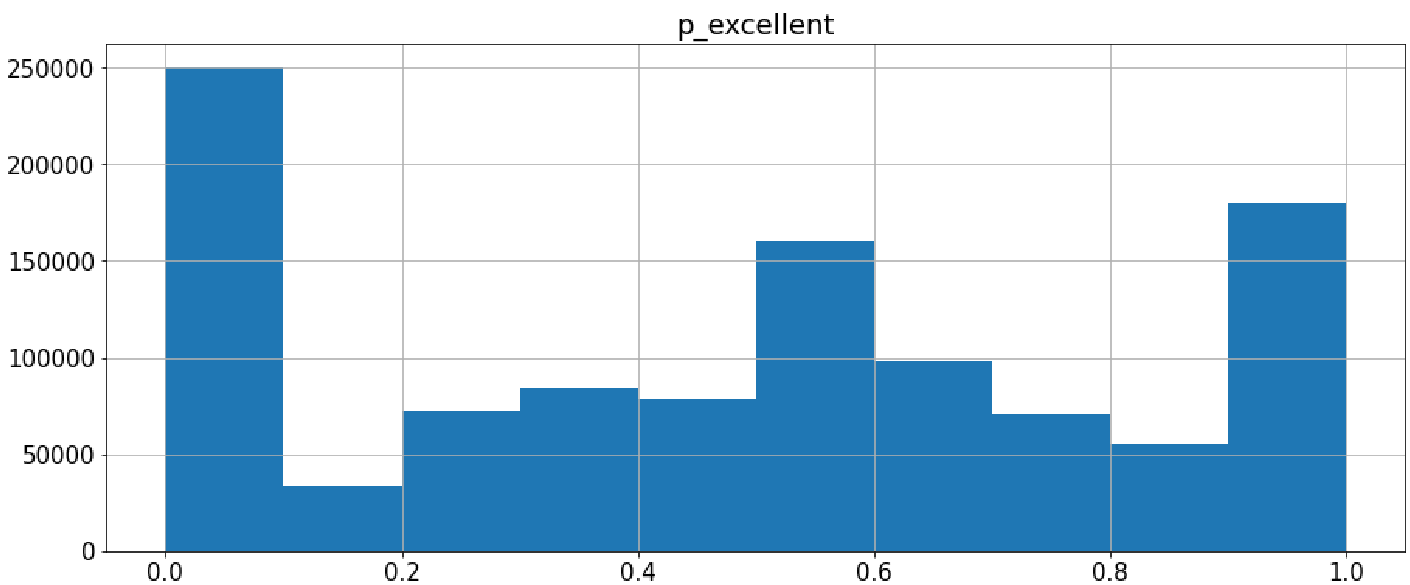


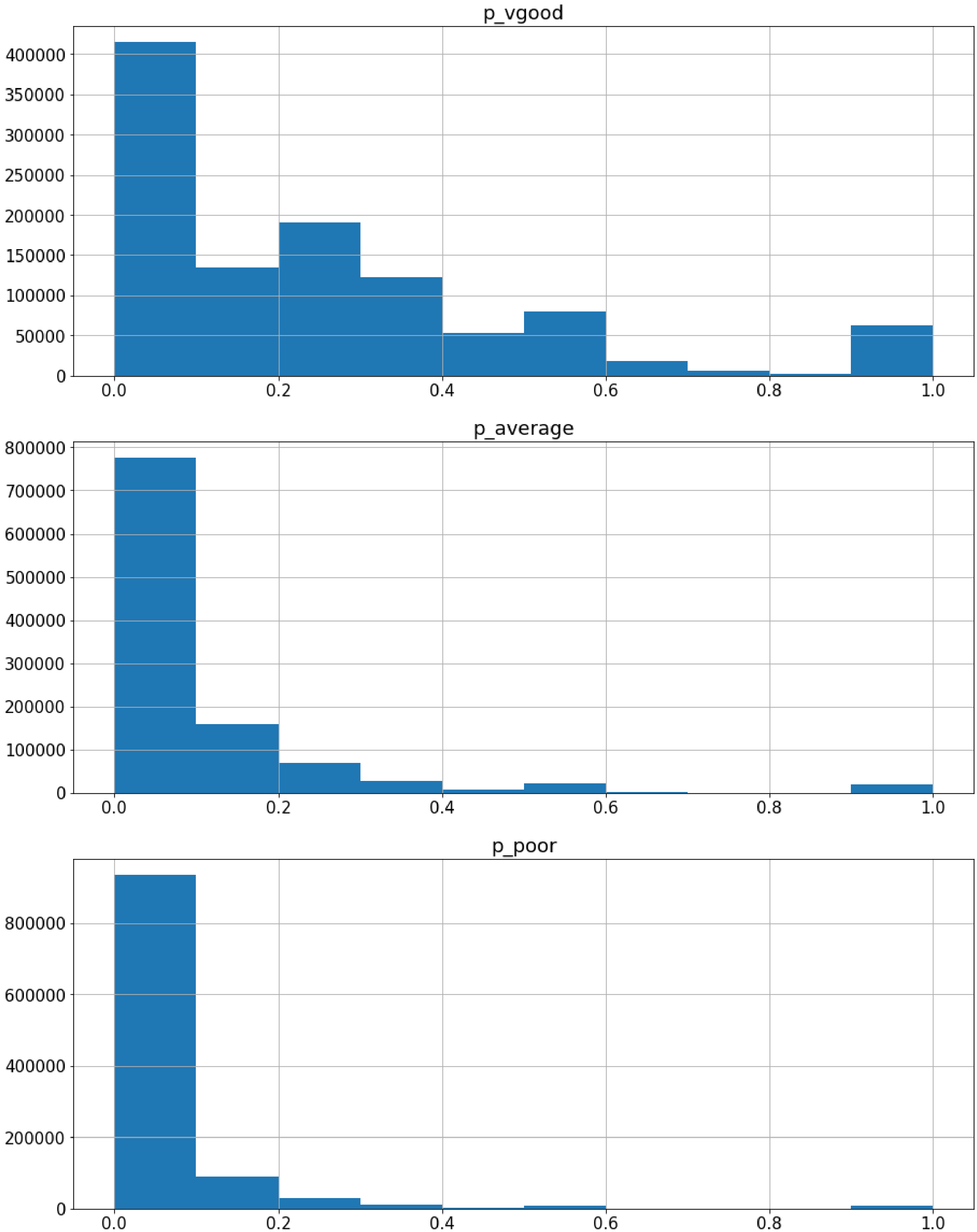
In [204…

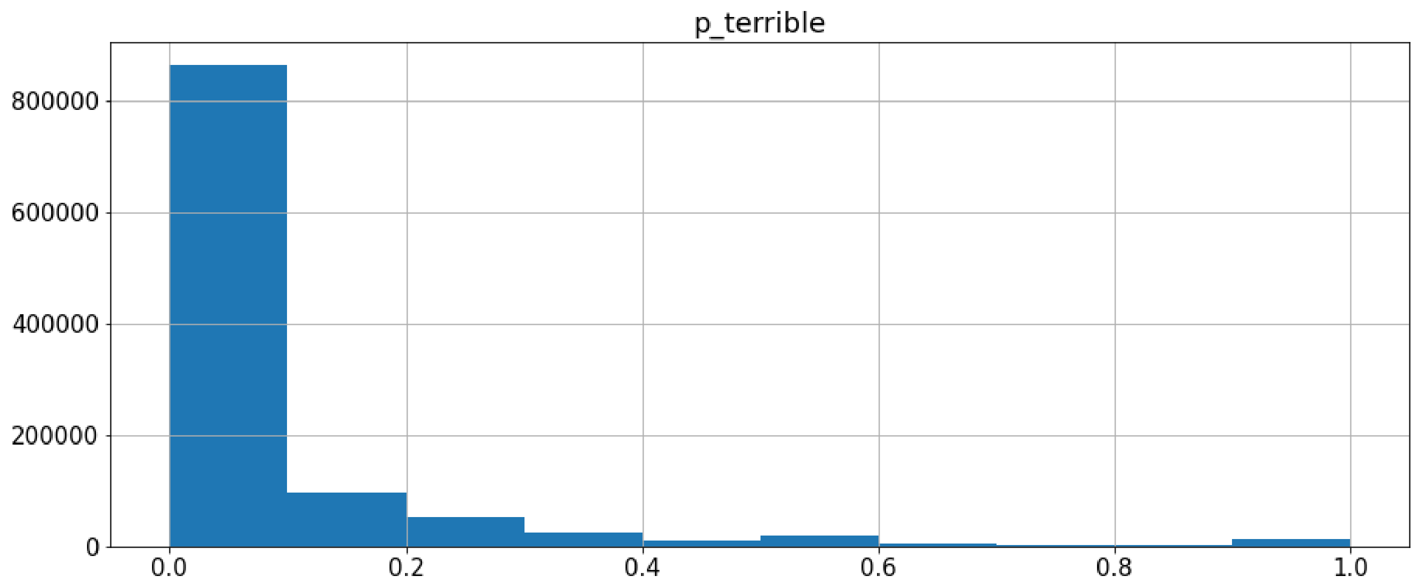
```
final_features.hist(column='p_excellent')
final_features.hist(column='p_vgood')
final_features.hist(column='p_average')
final_features.hist(column='p_poor')
final_features.hist(column='p_terrible')
plt.show()
```

Out[204…

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8b5f890d10>]],
      dtype=object)
```




p_vgood
0
50000
100000
150000
200000
250000
300000
350000
400000
0.0
0.2
0.4
0.6
0.8
1.0
p_average
0
100000
200000
300000
400000
500000
600000
700000
800000
0.0
0.2
0.4
0.6
0.8
1.0
p_poor
0
200000
400000
600000
800000
0.0
0.2
0.4
0.6
0.8
1.0

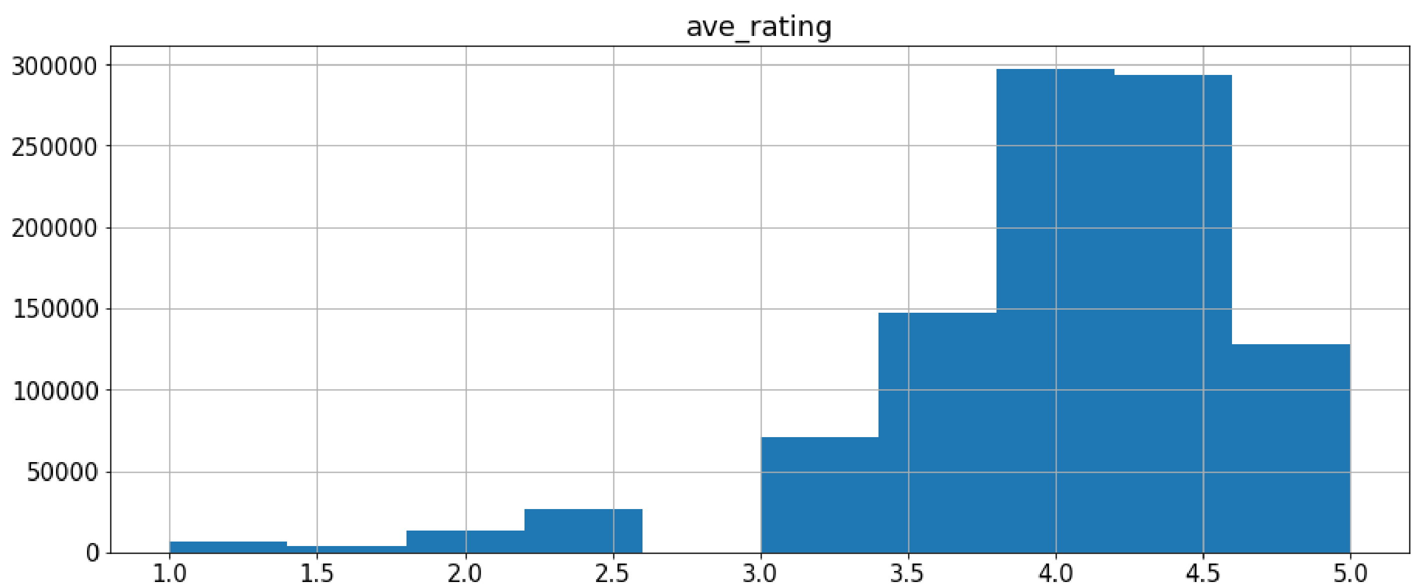


In [205…

```
final_features.hist(column='ave_rating')
plt.show()
```

Out[205…

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f8b5f8fa190>]],
      dtype=object)
```



In [279…

```
#encode binary values
final_features["claimed"] = pd.get_dummies(final_features["claimed"])["Claimed"]
final_features["veg"] = pd.get_dummies(final_features["veg"])["Y"]
final_features["vegan"] = pd.get_dummies(final_features["vegan"])["Y"]
final_features["gf"] = pd.get_dummies(final_features["gf"])["Y"]
final_features["awards"] = pd.get_dummies(final_features["awards"])["Y"]
```

2.2 Data Quality Issues and Data Quality Plan

Data Quality Issues:

The relevant features with highest percentage of missing values are price range (72% missing) and open hours per week (45% missing). As both these features are missing > 30% of their values and no other information is available to make reasonable estimates, these feature will be dropped.

As it is seen, the dataset does not have any features with a cardinality of 1. The rating features columns does not have too low cardinality for continous features. No issues with irregular cardinalities was found.

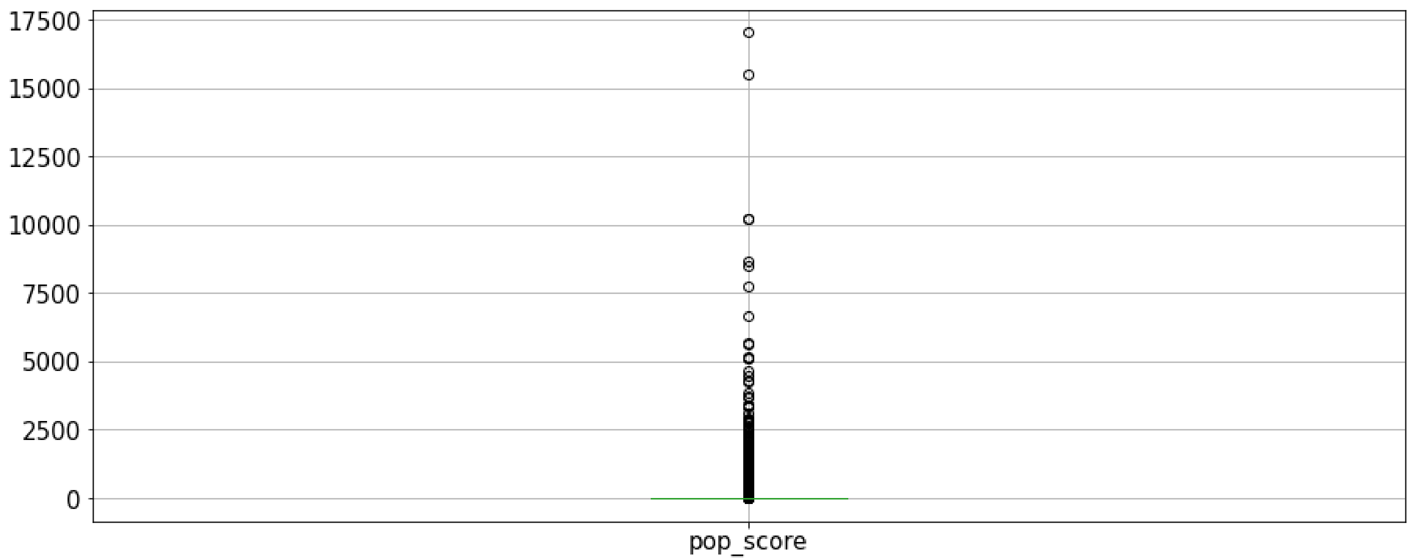
Outliers are suspected in the popularity score and are investigated below.

In [269…

```
final_features.boxplot('pop_score')
```

Out[269…

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8bad9a8510>
```



In [276…

```
# Code source: Tutorial

Q1 = final_features['pop_score'].quantile(0.25)
Q3 = final_features['pop_score'].quantile(0.75)

IQR = Q3-Q1
print(f"IQR = {Q3} - {Q1} = {IQR}")

outliers_df = final_features[(final_features['pop_score'] < (Q1 - 1.5 * IQR)) | (final_fea
print("Num of outliers: ", len(outliers_df))
mean = final_features['pop_score'].mean()
print("Mean", mean)
```

```
IQR = 3.1666666666666665 - 1.1324786324786325 = 2.034188034188034
Num of outliers:  128274
Mean 4.749224451041982
```

3. Data Preparation

Task 3.1

Removing features with highest percentage of missing values and awards column so that the missing values in awards would not impact on prediction of rating.

In [272…

```
final_features.drop("lower_price", axis=1, inplace=True)
final_features.drop("upper_price", axis=1, inplace=True)
final_features.drop("open_hours", axis=1, inplace=True)
```

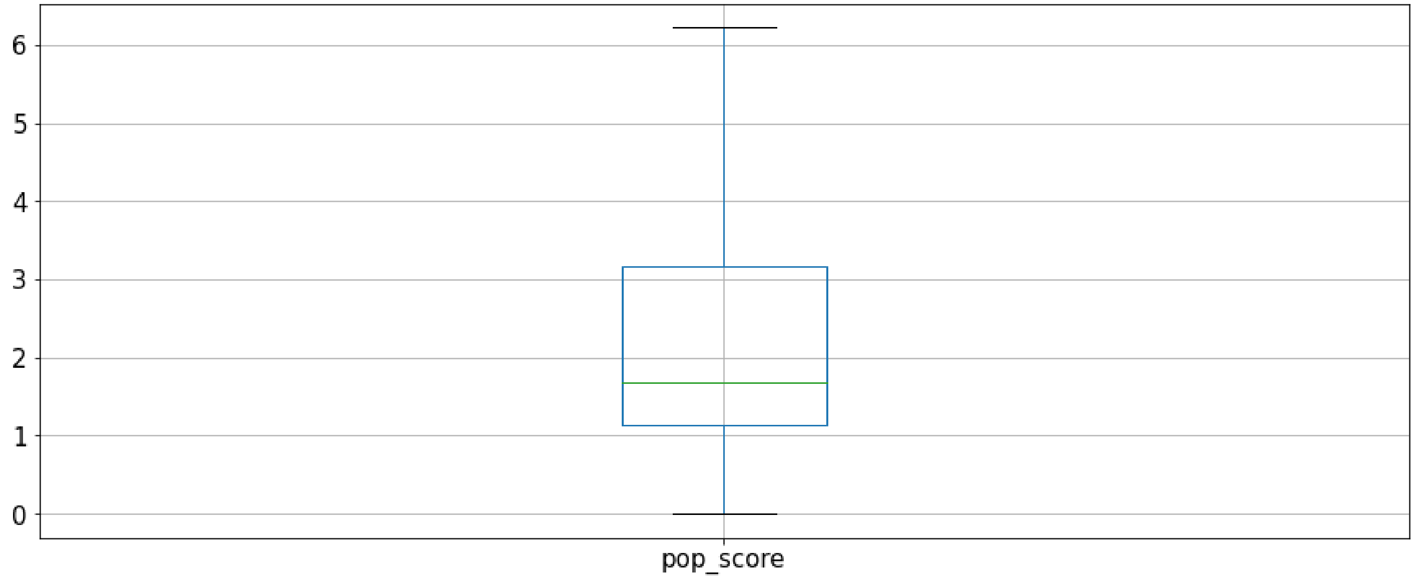
The outliers will be replaced the with mean value.

In [277…

```
final_features["pop_score"][outliers_df.index] = mean
final_features.boxplot('pop_score')
```

Out[277…

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b4ef7e310>
```

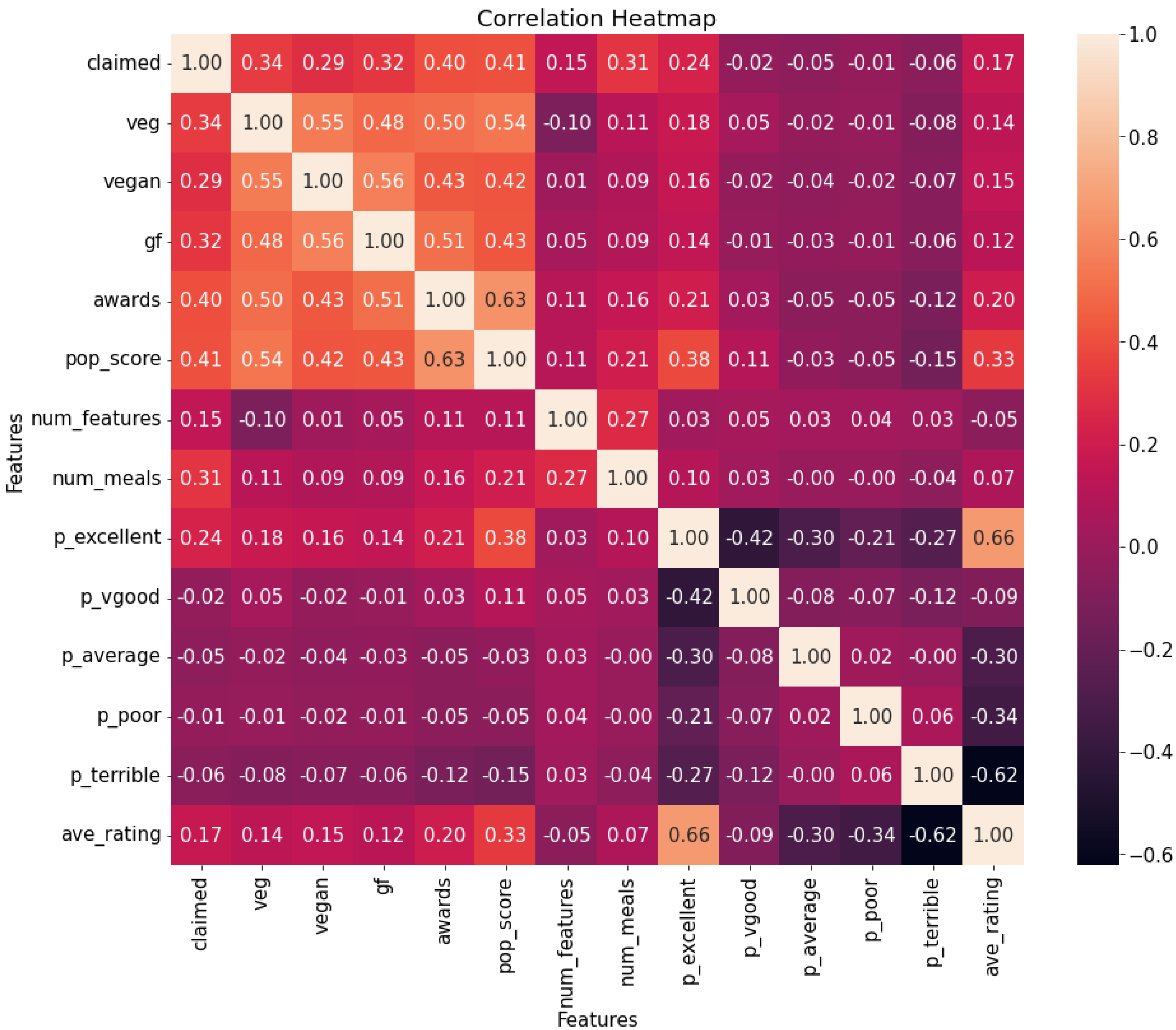


4. Data Insights

Note that the features for country, top cuisine and top tag were excluded due to too many categories.

In [280…

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(15, 12))
sns.heatmap(final_features.corr(), annot=True, fmt=".2f")
plt.title("Correlation Heatmap")
plt.xlabel("Features")
plt.ylabel("Features")
plt.show()
```



Task 4.1

Whether the restarurant is claimed and accessibility to special diet restrictions moderately correlate to the high rating target. The fact that the restaurant is claimed might indicate that the owners are more in touch with technology and able to use that to boost their business. Higher ratings could also be generated more easily by remaining accessible to people with dietary restrictions such as vegetarian, vegan, and gluten intolerant. The precense of an award and the percent of excellent reviews coorelate well to the target. The cause is intuitive as better restaurants will attract more awards, good reviews, and also higher rating. The popularity score also correlates highly to high rating, again as they are both increased for high quality restaurants.

Task 4.2

The number of meals correlates to the popularity rating, possibly due to increased service capabilities and therefore likely more customers. Awards, dietary restriction capabilities, claimed, and popularity are all highly correlated to one another. This could point to better menu diversity leads to more successful restaurants.

Task 4.3

The number of features does not seem to be correlated to any useful features. The percent of average and poor reviews is not a very useful feature either.

References

- https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.
- https://www.askpython.com/python/examples/heatmaps-in-python
- In-class Tutorial Material: https://dal.brightspace.com/d2l/le/content/232269/Home

In [ ]:

```

```