notebook

March 22, 2023

1 Data Scientist Professional Practical Exam Submission

Use this template to write up your summary for submission. Code in Python or R needs to be included.

2 Data Validation

Importing our data to clean, I write the following code:

[37]:
```
import pandas as pd

df = pd.read_csv('recipe_site_traffic_2212.csv')
print(df.shape)
```

```
(947, 8)
```

[38]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 947 entries, 0 to 946
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   recipe        947 non-null    int64  
 1   calories      895 non-null    float64
 2   carbohydrate  895 non-null    float64
 3   sugar         895 non-null    float64
 4   protein       895 non-null    float64
 5   category      947 non-null    object 
 6   servings      947 non-null    object 
 7   high_traffic  574 non-null    object 
dtypes: float64(4), int64(1), object(3)
memory usage: 59.3+ KB
None
```

[39]:
```
print(df.isna().sum())
```

```
recipe            0
calories         52
carbohydrate     52
sugar            52
protein          52
category          0
servings          0
high_traffic    373
dtype: int64
```

Our dataset contains **8 columns** and **947 rows** without any cleaning. Looking at each column:

- Recipe column contains 947 unique integers, fine to use as a unique identifier.
- Category column contains 11 different categories, not 10. Inspecting this:

[40]:
```
print(df.category.unique())
df.category = df.category.replace('Chicken Breast', 'Chicken')
print(df.category.unique())
```

```
['Pork' 'Potato' 'Breakfast' 'Beverages' 'One Dish Meal' 'Chicken Breast'
 'Lunch/Snacks' 'Chicken' 'Vegetable' 'Meat' 'Dessert']
['Pork' 'Potato' 'Breakfast' 'Beverages' 'One Dish Meal' 'Chicken'
 'Lunch/Snacks' 'Vegetable' 'Meat' 'Dessert']
```

All matches up except 'Chicken Breast' which is replaced with with 'Chicken' to give the correct amount of categories. - Servings column is not numeric with '4 as snack' and '6 as snack' as values. I decide to remove these values as they are only 3 rows in Servings which have these and they is no relation to convert these to regular servings.

[41]:
```
df = df[~df['servings'].isin(['6 as a snack','4 as a snack'])]
df['servings'] = df['servings'].astype('int64')
```

- High traffic column contains 372 null values and the rest of the values are labeled 'High'. I replace 'High' with True, and null values with False.

[42]:
```
df.high_traffic = df.high_traffic.fillna(False)
df.high_traffic = df.high_traffic.replace('High', True)
df['high_traffic'] = df['high_traffic'].astype('bool')
```

- Calories column is numeric with 52 null values.
- Carbohydrate column is numeric with 52 null values.
- Sugar column is numeric with 52 null values.
- Protein column is numeric with 52 null values. As these null values all share the same rows, I decide to remove all these null values.

[43]:
```
df = df.dropna()
print(df.shape)
```

```
(892, 8)
```

After the data validation, the dataset contains **892 rows and 8 columns**.

3 Exploratory Analysis

Before making a model, lets have a look to see if each variable can affect if the traffic will be high or not. ## Nutrients vs traffic

From **Graph 1**, we compare the carbohydrate, protein and sugar column with the high_traffic column. We omit outliers as we have a large enough dataset, and our outliers obstruct our view of the boxplots. Looking at Graph 1, we can come to a conclusion that they is a strong enough evidence to show that any of the nutrients' amounts affects the traffic.

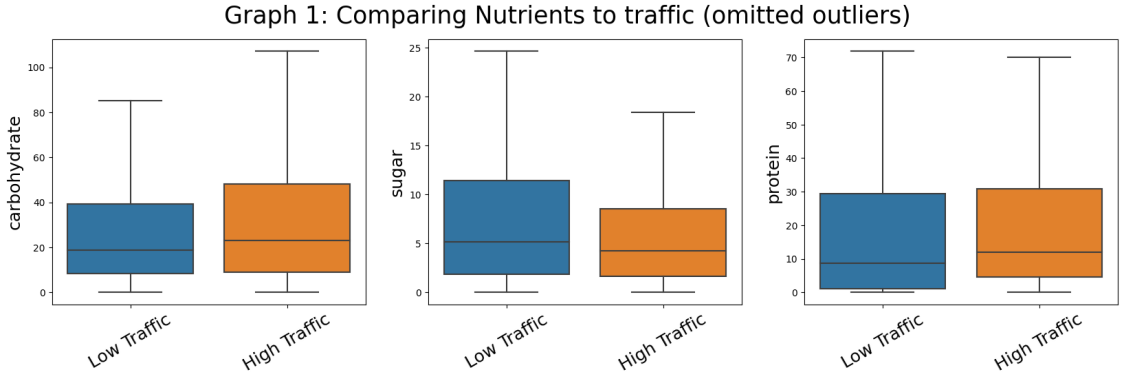
[44]:
```
import matplotlib.pyplot as plt
import seaborn as sns

cols = ['carbohydrate','sugar','protein',]
fig, axs = plt.subplots(1, 3 , figsize=(20,5))

for i, col in enumerate(cols):
    sns.boxplot(x='high_traffic', ax = axs[i], y=col, data=df,showfliers=False)
    axs[i].set_xticklabels(['Low Traffic', 'High Traffic'], rotation=30,␣
 ↪fontsize=18)
    axs[i].set_ylabel(col, fontsize=18)
    axs[i].set_xlabel('')


fig.suptitle('Graph 1: Comparing Nutrients to traffic (omitted outliers)',␣
 ↪fontsize=25)
```

[44]:
```
Text(0.5, 0.98, 'Graph 1: Comparing Nutrients to traffic (omitted outliers)')
```



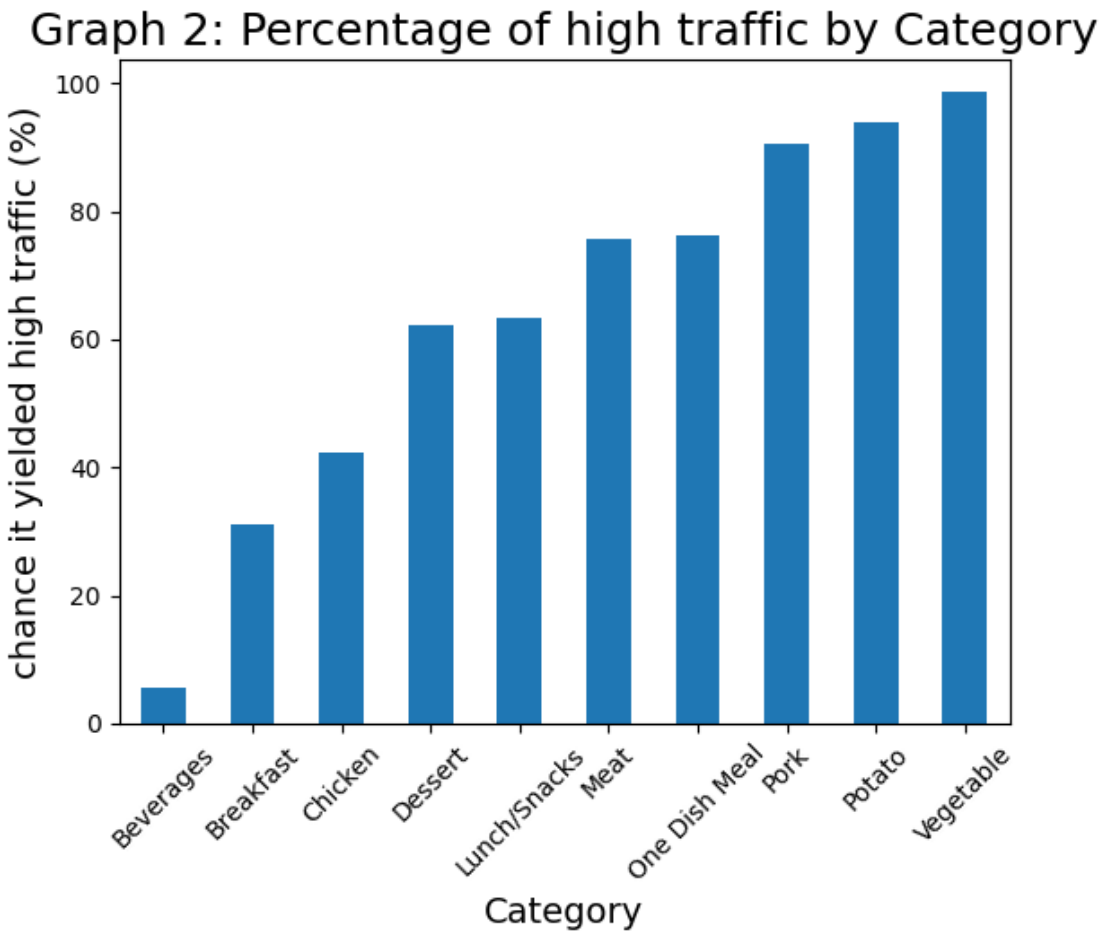
3.1 Food Category vs traffic

From **Graph 2**, I compare the Food Category with the traffic data. I group the data by category and look at the percentage of high traffic occurs from each category. This graph does not take account of the count of each category, but I have looked through the data and found that each category has atleast 67 counts showing that this graph is suitable for use. We can see that the

best perfoming category is Vegetable with 98.7% of Vegetable recipes having high traffic. This is followed by Potato (94.0%) and Pork (90.4%). The worst category is Beverages with 5.4% of Beverages recipes being high traffic.

[45]:
```
vote_perc = df[['category','high_traffic']].groupby('category')['high_traffic'].
  ↪mean()
vote_perc = vote_perc*100
vote_perc.plot.bar()
plt.xlabel('Category',fontsize=14)
plt.xticks(rotation=45)
plt.ylabel('chance it yielded high traffic (%)', fontsize=14)
plt.title('Graph 2: Percentage of high traffic by Category',fontsize=18)
```

[45]:
```
Text(0.5, 1.0, 'Graph 2: Percentage of high traffic by Category')
```

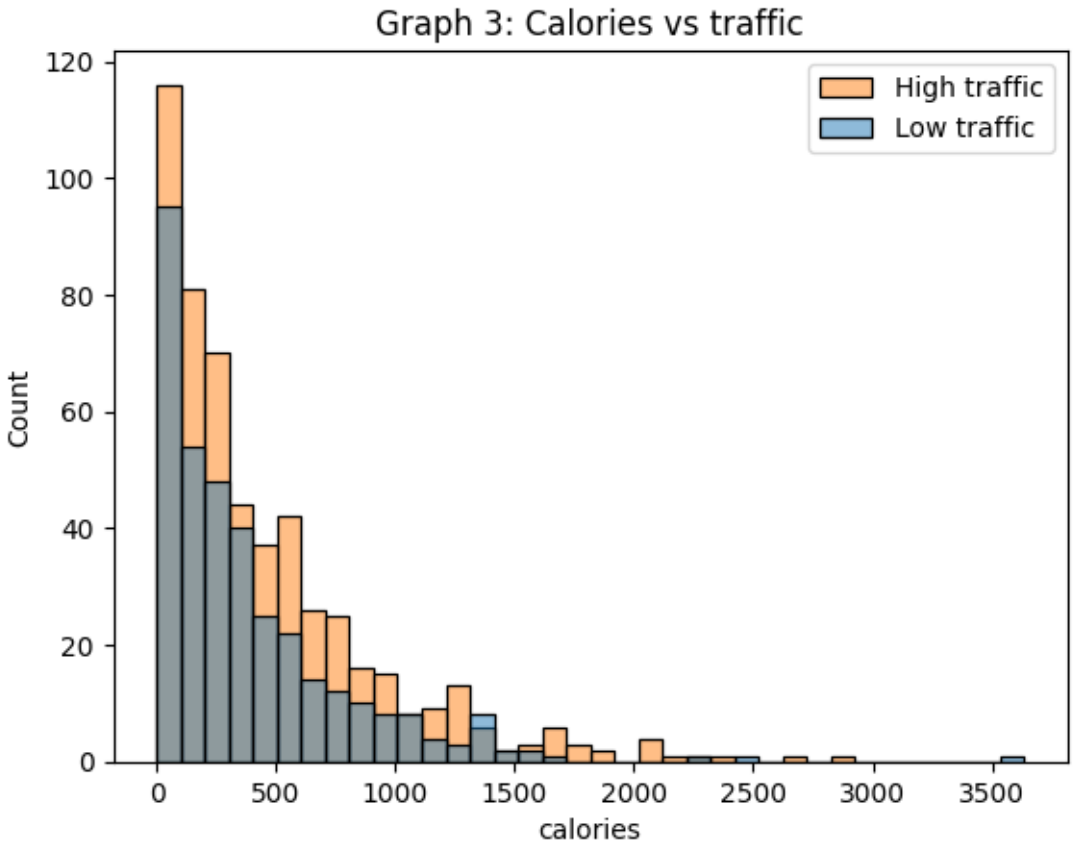


3.2 Calories vs traffic

From **Graph 3**, I compare the calories and traffic columns. As we can see, they is no significant correlation that the calorie amount affects the traffic.

[46]:
```
g3 = sns.histplot(x='calories', hue='high_traffic', data=df)
plt.legend(['High traffic', 'Low traffic'])
g3.set_title('Graph 3: Calories vs traffic')
```

[46]:
```
Text(0.5, 1.0, 'Graph 3: Calories vs traffic')
```



3.3 Servings vs Traffic

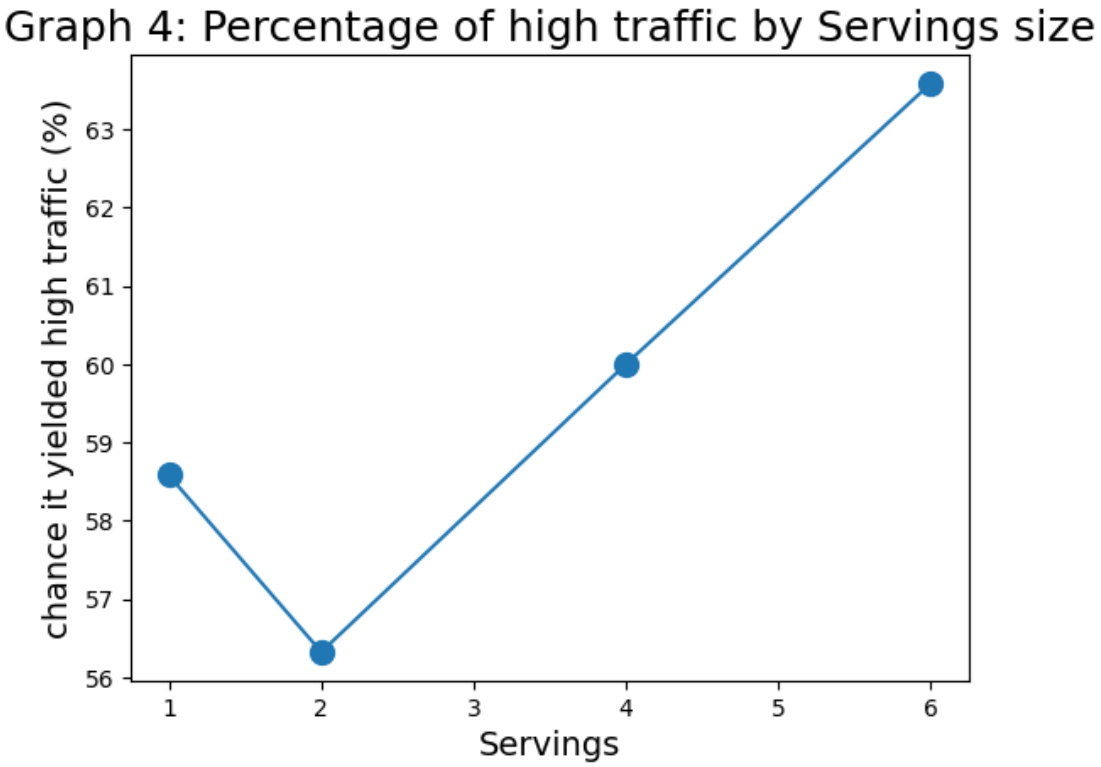
[ ]:
```
ntpaint
```

Again with **Graph 4**, They is not a significant difference to say that serving size affects if the traffic is high.

[47]:
```
vote_perc = df[['servings','high_traffic']].groupby('servings')['high_traffic'].
  ↪mean()
vote_perc = vote_perc*100
```

```
vote_perc.plot.line(marker='.', markersize=20)
plt.xlabel('Servings',fontsize=14)
plt.xticks(rotation=0)
plt.ylabel('chance it yielded high traffic (%)', fontsize=14)
plt.title('Graph 4: Percentage of high traffic by Servings size',fontsize=18)
```

[47]: `Text(0.5, 1.0, 'Graph 4: Percentage of high traffic by Servings size')`



3.4 Model Development

As we are trying to predict whether traffic will be high or not, this is a **classification problem**. We will use classification models such as **Logisitic regression (as the base modeL)** since our output is binary, **Decision trees and Random Forest (to compare)** as it works well with with mixed data types. First I will look at logisitic Regression and Decision Trees. Pre-processing our data, my code is:

[48]:
```
from sklearn.model_selection import train_test_split
df['high_traffic'] = df['high_traffic'].replace({True:1,False:0}) #change true␣
 ↪and false to binary
y = df['high_traffic']
df2 = df.drop(columns=['recipe','high_traffic']) #dropping columns for X data
X = pd.get_dummies(df2)
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3,␣
↪random_state=42) #split data 30% test, 70% train
```

To analyse the **logisitic regression model & decision tree model**, my code is:

[49]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

logreg = LogisticRegression()
logreg.fit(X_train,y_train)
y_pred_log =logreg.predict(X_test)

clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred_clf = clf.predict(X_test)
```

To look at how well each model performs, I use a classification report. For **logisitic regression**, the precision is 0.68 for low traffic and 0.80 for high traffic.

[50]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
print(classification_report(y_test, y_pred_log))
```

```
              precision    recall  f1-score   support

           0       0.68      0.69      0.69       104
           1       0.80      0.79      0.80       164

    accuracy                           0.75       268
   macro avg       0.74      0.74      0.74       268
weighted avg       0.75      0.75      0.75       268
```

And for **decision tree model**, the precision for low traffic is 0.57 and for high traffic is 0.78. **Logistic regression** can predict better than **Decision Tree model** but both these models **fails** to predict the traffic outcome 80% of the time here.

[51]:
```
print(classification_report(y_test, y_pred_clf))
```

```
              precision    recall  f1-score   support

           0       0.57      0.70      0.63       104
           1       0.78      0.66      0.71       164

    accuracy                           0.68       268
   macro avg       0.67      0.68      0.67       268
weighted avg       0.70      0.68      0.68       268
```

As both our models are weak, I decide to use the **random forest model**.

[52]:
```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(n_estimators=100, random_state=0)
rf.fit(X, y)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       104
           1       1.00      1.00      1.00       164

    accuracy                           1.00       268
   macro avg       1.00      1.00      1.00       268
weighted avg       1.00      1.00      1.00       268
```

This **random forest model** can predict if the traffic will be high or low 100% of the time which is great for us as it predicts correctly over 80% of the time.

[53]:
```
print(confusion_matrix(y_test,y_pred_rf))
```

```
[[104   0]
 [  0 164]]
```

3.5 Business Metrics

Since our goal is to increase the number of subscriptions which is correlated to the percentage of high traffic on the website, I would recommend we use the **percentage of high traffic recipes** as our metric. In our dataset, 59.8% of our recipes have high traffic, therefore if we can correctly increase traffic by displaying a certain recipe, it is a good sign to achieve our goal. Our random forest model can predict if a recipe will have high traffic well, with 100% accuracy with our test data.

3.6 Recommendation

For the future, I would recommend we can focus on the following steps:

- Not to display Beverage recipes.
- Use our model (Random Forest) to help predict which recipes will return high traffic.
- Be aware that chicken and breakfast recipes may reduce traffic, but more analysis on this must be done to see why this is the case.
- Promote pork, potato and vegetable dishes as they return higher traffic.
- Serving size may affect traffic but more data needs to be gathered with more analysis done to see how nutrients and calories intertwine with serving size.
- Maybe collect user data to see why they are interested in certain recipes?
- Improve data quality - Potatoes are vegetables but are classed as different categories. What's in a one dish meal? what's in lunch/snacks? These could contain meat or vegetables etc.

- Servings sizes only 1,2,4 and 6. Could maybe analyse different serving sizes.