In [1]:
```
import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import numpy as np
from plotnine import *

from sklearn.tree import DecisionTreeClassifier # Decision Tree
from sklearn.linear_model import LogisticRegression # Logistic Regress
ion Model
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
, CategoricalNB # Decision Tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.preprocessing import StandardScaler #Z-score variables

from sklearn.model_selection import train_test_split # simple TT split
cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.model_selection import cross_val_score # cross validation
metrics
from sklearn.model_selection import cross_val_predict # cross validati
on metrics
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix

from sklearn.model_selection import GridSearchCV

%precision %.7g
%matplotlib inline
```

Part I

In [2]:
```
aye = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392ParlettPelleriti/master/Data/burgersOrPizza.csv")
aye.head()
```

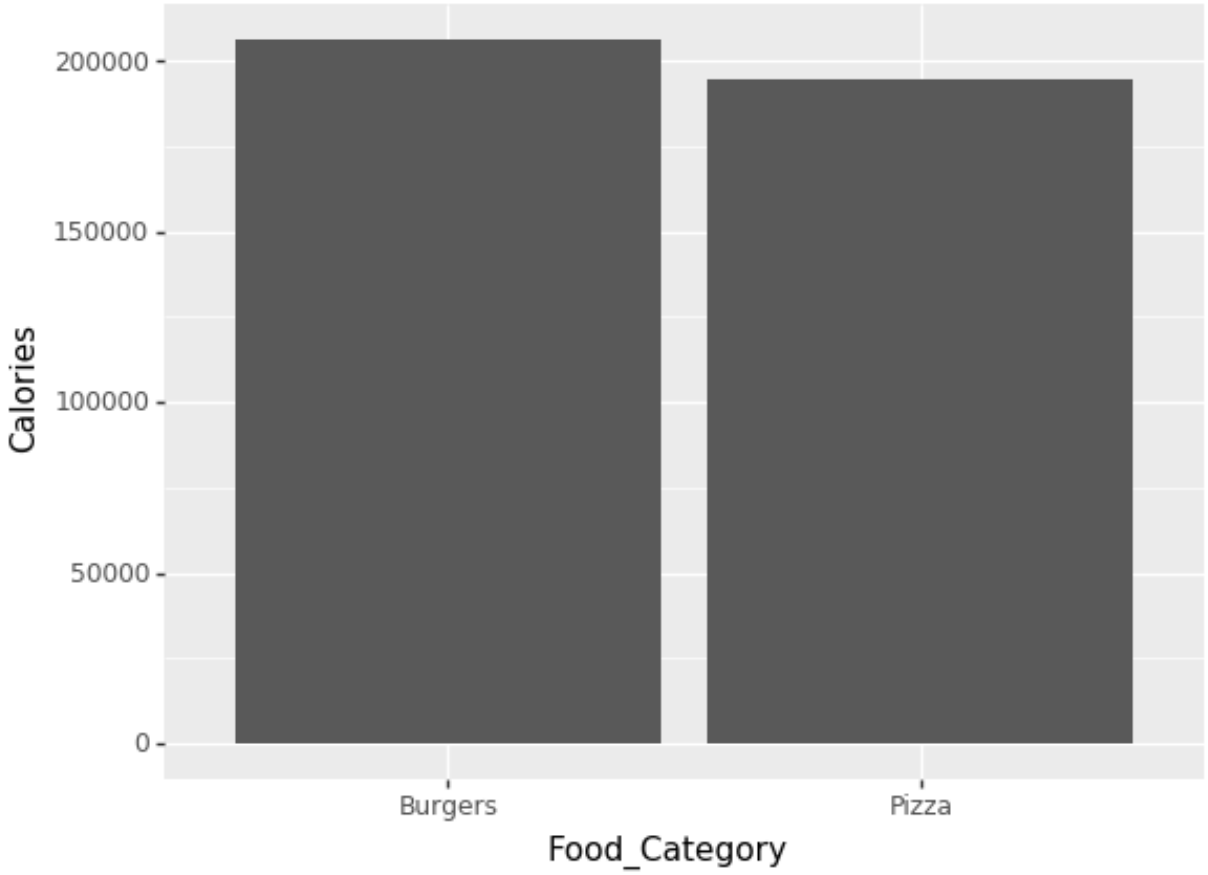
Out[2]:

	Item_Name	Item_Description	Food_Category	Calories	Total_Fat	Saturated_Fat	Trans_Fat
0	Chicken n Cheese Slider	Chicken n Cheese Slider on Mini Bun w/ Chicken...	Burgers	290.0	12.0	3.5	0.0
1	Corned Beef n Cheese Slider	Corned Beef n Cheese Slider on Mini Bun w/ Cor...	Burgers	220.0	9.0	3.5	0.0
2	Ham n Cheese Slider	Ham n Cheese Slider on Mini Bun w/ Roast Ham &...	Burgers	230.0	9.0	3.5	0.0
3	Jalapeno Roast Beef n Cheese Slider	Jalapeno Roast Beef n Cheese Slider on Mini Bu...	Burgers	240.0	11.0	4.5	0.0
4	Roast Beef n Cheese Slider	Roast Beef n Cheese Slider on Mini Bun w/ Roas...	Burgers	240.0	11.0	4.5	0.0

5 rows × 25 columns

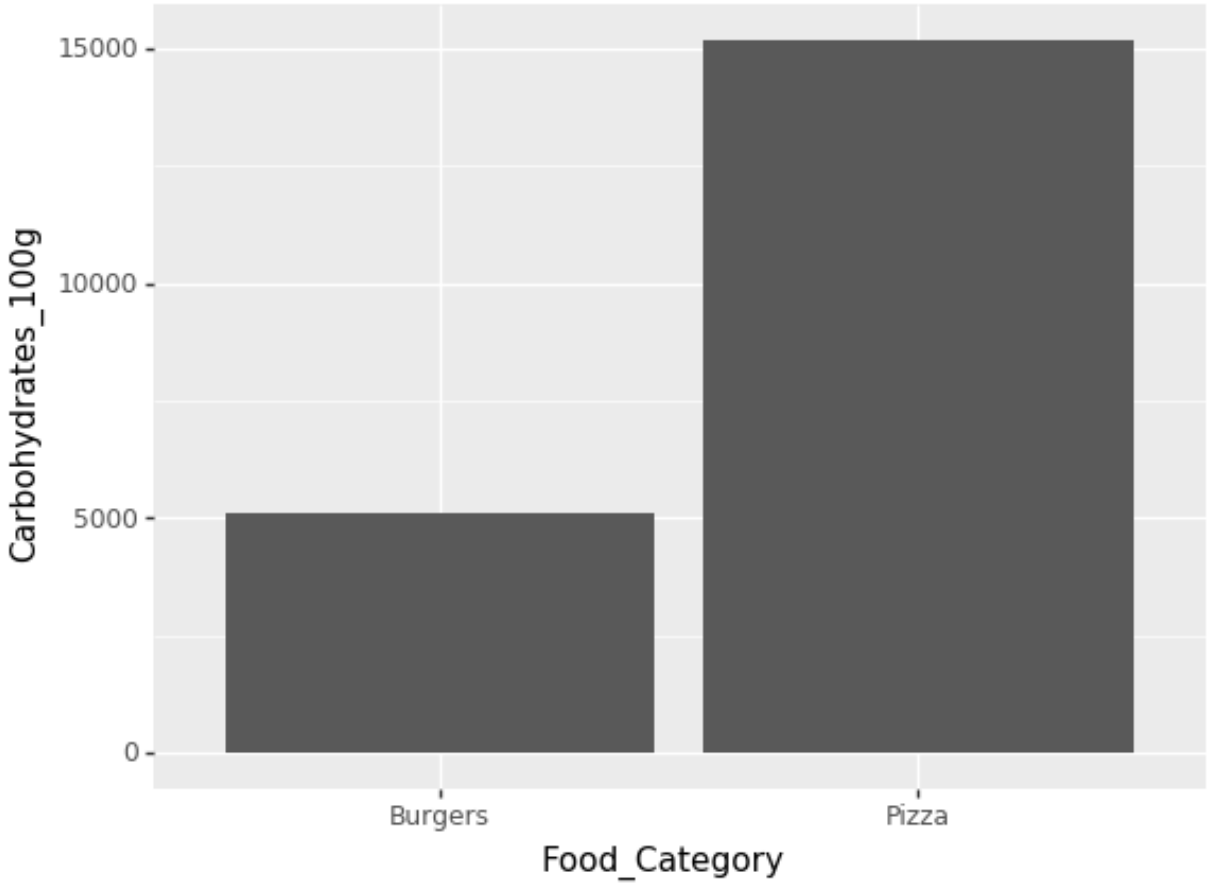
1. Explore Data (with ggplot)

In [27]:
```
(ggplot(aye, aes(x = "Food_Category", y = "Calories")) + geom_bar(stat
= "identity"))
```



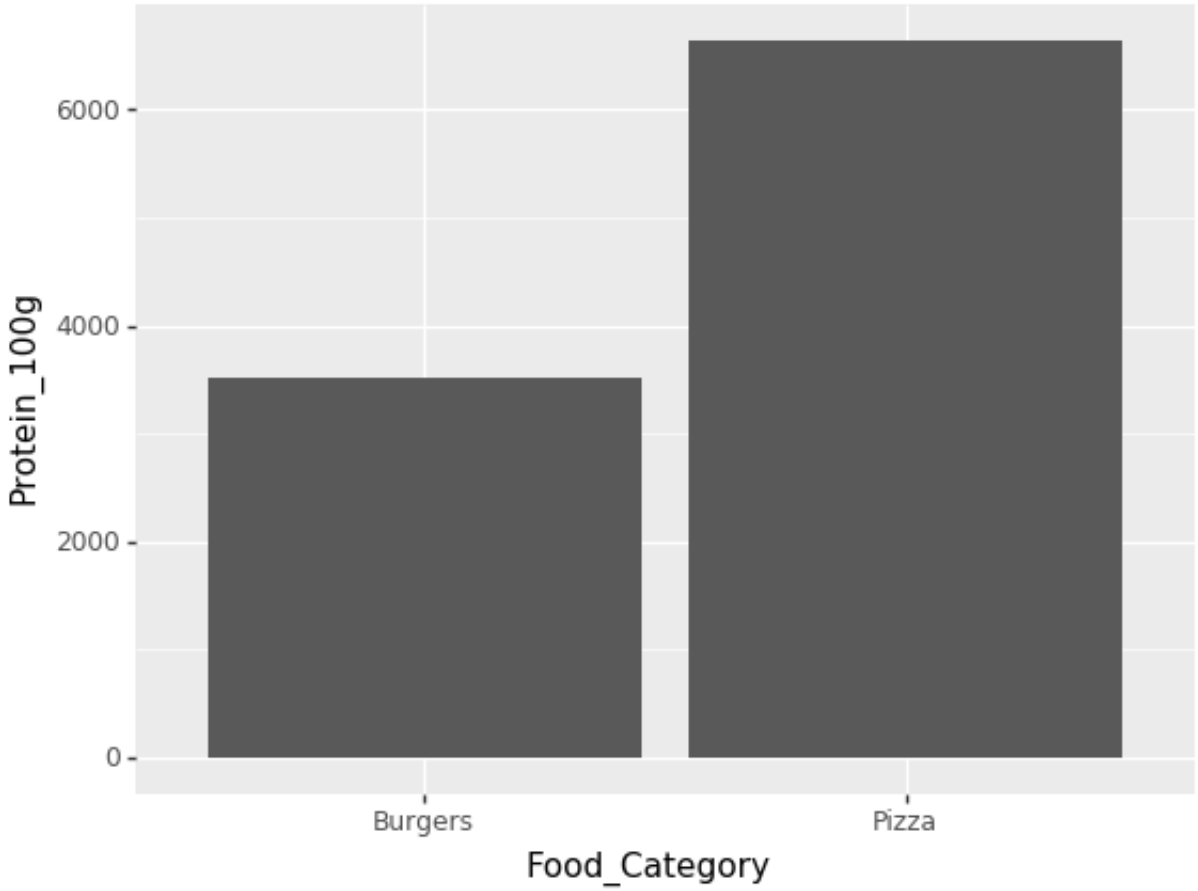
Out[27]: <ggplot: (307672741)>

In [28]:
```
(ggplot(aye, aes(x = "Food_Category", y = "Carbohydrates_100g")) + geo
m_bar(stat = "identity"))
```



Out[28]: <ggplot: (307202857)>

In [29]:
```
(ggplot(aye, aes(x = "Food_Category", y = "Protein_100g")) + geom_bar(
stat = "identity"))
```

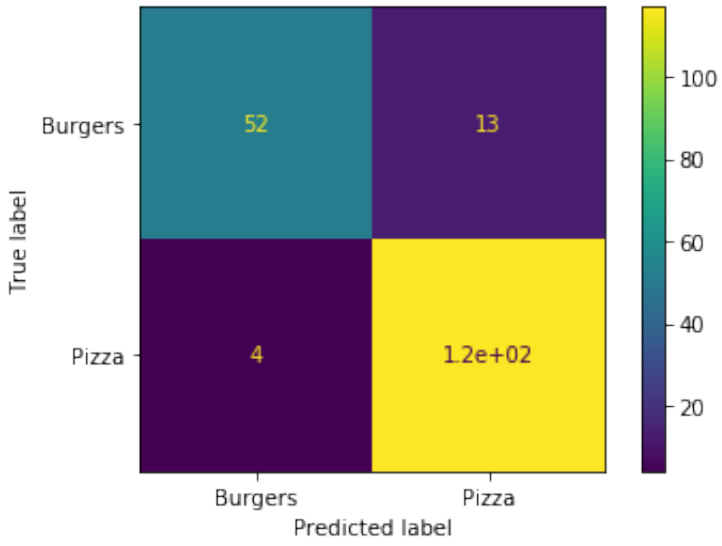


Out[29]: <ggplot: (276454737)>

Model 1

In [33]:
```
predictors = ["Calories", "Carbohydrates_100g", "Protein_100g"]

X = aye[predictors]
y = aye["Food_Category"]

# split into training and test
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0
.2)

#standardize
z = StandardScaler()
X_train = z.fit_transform(X_train)
X_test = z.transform(X_test)

# create model
knn2 = KNeighborsClassifier()

# choose potential values of k
ks = {"n_neighbors": range(1,30)}

# use grid search to find best parameters
grid = GridSearchCV(knn2,ks, scoring = "accuracy", cv = 30)

knnmod = grid.fit(X_train, y_train)

plot_confusion_matrix(knnmod, X_test, y_test)
```

Out[33]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x
124bb9c90>
```



In [34]:
```
knnmod.best_estimator_.get_params()["n_neighbors"]
```

Out[34]: 3

In [35]:
```
knnmod.score(X_test,y_test)
```

Out[35]: 0.9086021505376344

2. Explain which variables you're using to predict the outcome

The variables I used to predict the outcome are calories, carbohydrates, and protein. I used these variables to predict the outcome because usually burgers tend to have more calories and protein compared to pizza. Whereas, a pizza would have more carbohydrates compared to a hamburger.

3. Explain which model validation technique you're using and why.

The model validation technique I used is hyper parameter grid searching. This helps me find the best parameter.

4. Explain why you did or did not choose to standardize your continuous variables.

I chose to standardize my continuous variable because if the variables are measured in different scales it can impact the way they’re weighed when measuring variation.

5. Evaluate how the model performed. Explain.

My model performed well and it shows a 90.86% accuracy score. Looking at the confusion matrix, the off-diagonal it was able to predict most of it correctly.

Model 2

In [21]:
```
X = aye[["Calories","Carbohydrates_100g","Protein_100g"]]
y = aye["Food_Category"]

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0
.2)
zscore = StandardScaler()
zscore.fit(X_train)

X_train = zscore.transform(X_train)
X_test = zscore.transform(X_test)

tree = DecisionTreeClassifier()
tree.fit(X_train, y_train)
```

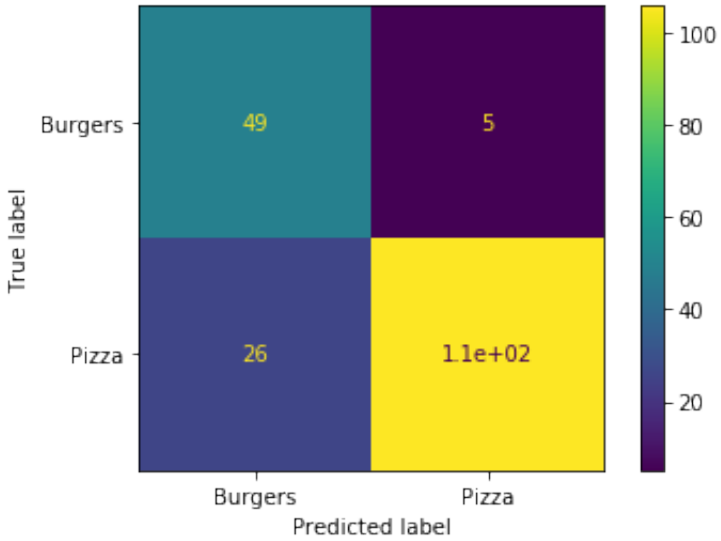
Out[21]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='
gini',
                       max_depth=None, max_features=None, max_leaf_n
odes=None,
                       min_impurity_decrease=0.0, min_impurity_split
=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprec
ated',
                       random_state=None, splitter='best')
```

In [22]:
```
y_pred = tree.predict(X_test)

accuracy_score(y_pred, y_test)
```

Out[22]:
```
0.8333333333333334
```

In [23]:
```
plot_confusion_matrix(tree,X_test,y_test)
```

Out[23]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x
11f755dd0>
```



In [24]:
```
tree.get_depth()
```

Out[24]:
```
17
```

In [25]:
```
tree.get_n_leaves()
```

Out[25]:
```
124
```

In [26]:
```
tree = DecisionTreeClassifier(max_depth = 5)
tree.fit(X_train, y_train)
```

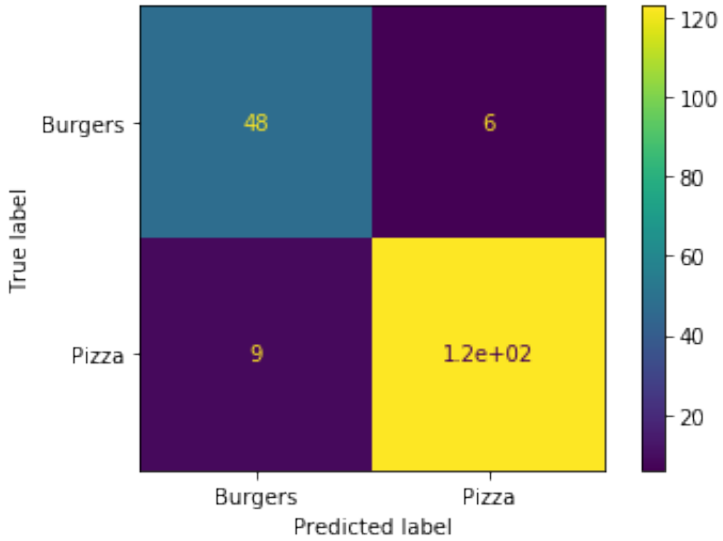
Out[26]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='
gini',
                       max_depth=5, max_features=None, max_leaf_node
s=None,
                       min_impurity_decrease=0.0, min_impurity_split
=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprec
ated',
                       random_state=None, splitter='best')
```

In [27]:
```
y_pred = tree.predict(X_test)

accuracy_score(y_pred, y_test)
```

Out[27]:
```
0.9193548387096774
```

In [28]:
```
plot_confusion_matrix(tree,X_test,y_test)
```

Out[28]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x
11f98b390>
```



2. Explain which variables you're using to predict the outcome.

The variables I used to predict the outcome are calories, carbohydrates, and protein. I used these variables to predict the outcome because usually burgers tend to have more calories and protein compared to pizza. Whereas, a pizza would have more carbohydrates compared to a hamburger.

3.Explain which model validation technique you're using and why.

The model validation technique I used is train split test. I used this technique because our data includes different nutrition facts of pizza and burgers. The train split test technique will help boost the algorithm performance. We will train the models, each time using one variable for testing and the other variables for training. Then, we will be able to evaluate better our algorithm and be able to build our model.

4. Explain why you did or did not choose to standardize your continuous variables.

I chose to standardize my continuous variable because if the variables are measured in different scales it can impact the way they’re weighed when measuring variation.

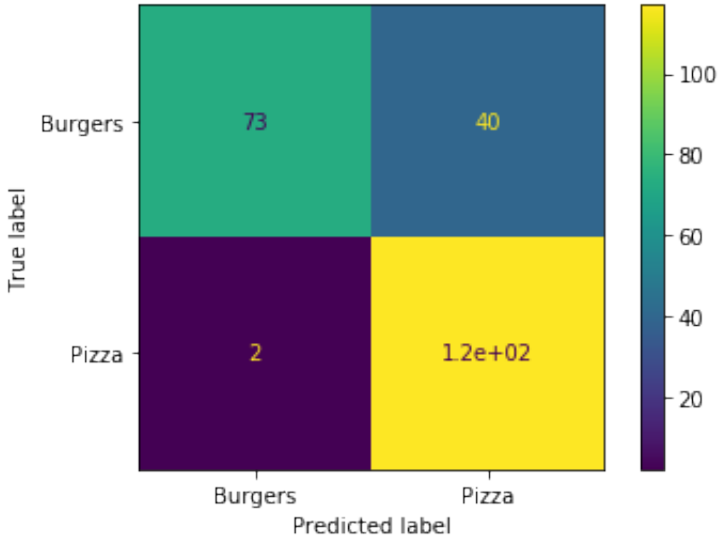
5. Evaluate how the model performed. Explain

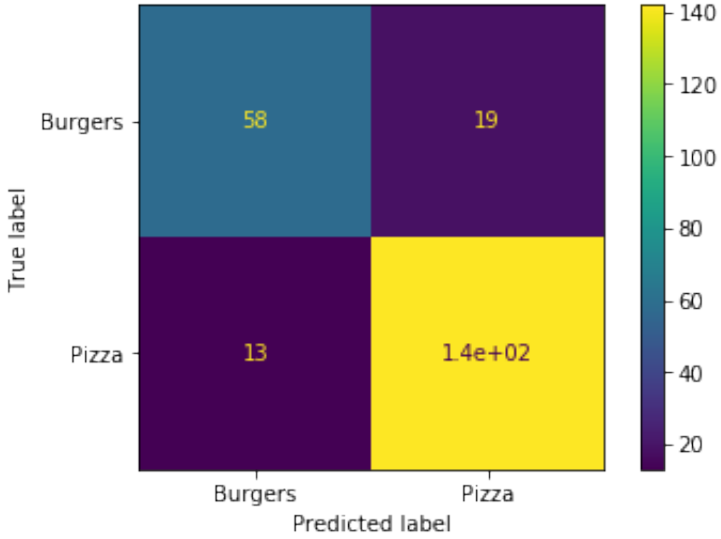
My model performed well and it shows a 91.94% accuracy score. Looking at the confusion matrix, the off-diagonal it was able to predict most of it correctly.

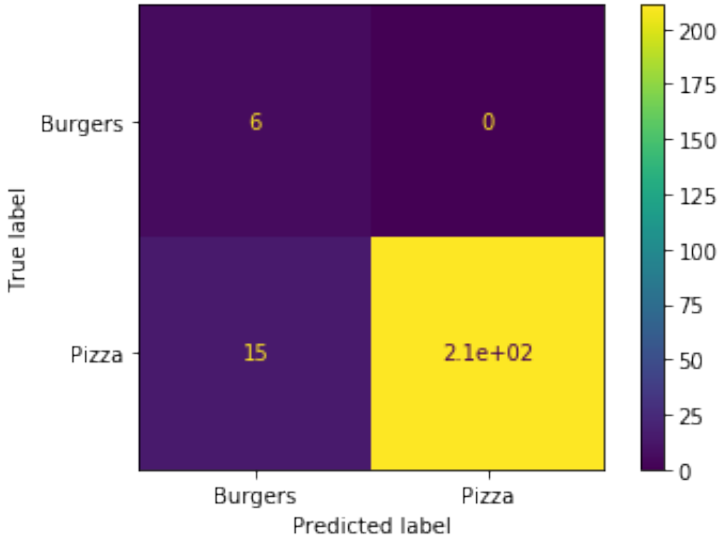
Model 3

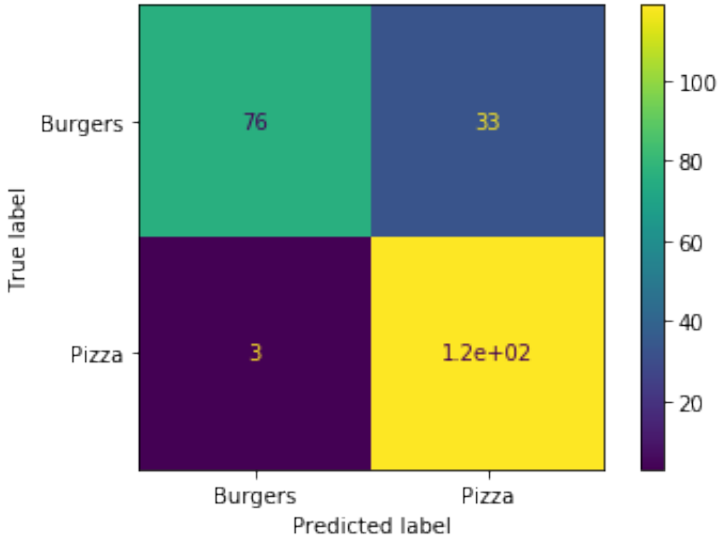
In [5]:
```
pred = ["Calories", "Carbohydrates_100g", "Protein_100g"]
X = aye[pred]
y = aye["Food_Category"]

pizza = np.where(y == 'Pizza')[0]
burger = np.where(y == 'Burgers')[0]

pizzaDownSample = np.random.choice(pizza, size = len(burger), replace
= False)

downSampledData = np.concatenate([burger,pizzaDownSample])
downSampledData

X_d = aye.iloc[downSampledData,][pred]
y_d = aye.iloc[downSampledData,]["Food_Category"]

X_d.shape

kf = KFold(n_splits = 4)
nb = GaussianNB()
acc = []
for train, test in kf.split(X,y):
    X_train = X.iloc[train]
    X_test = X.iloc[test]
    y_train = y[train]
    y_test = y[test]

    nb.fit(X_train,y_train)
    acc.append(nb.score(X_test,y_test))
    plot_confusion_matrix(nb, X_test,y_test)

print(acc)
print(np.mean(acc))
```

```
[0.8189655172413793, 0.8620689655172413, 0.9353448275862069, 0.84415
58441558441]
0.8651337886251679
```


Burgers
Pizza
True label
73
40
2
1.2e+02
Burgers
Pizza
Predicted label
100
80
60
40
20


Burgers
Pizza
True label
58
19
13
1.4e+02
Burgers
Pizza
Predicted label
140
120
100
80
60
40
20


Burgers
Pizza
True label
6
0
15
2.1e+02
Burgers
Pizza
Predicted label
200
175
150
125
100
75
50
25
0



2. Explain which variables you're using to predict the outcome.

The variables I used to predict the outcome are calories, carbohydrates, and protein. I used these variables to predict the outcome because usually burgers tend to have more calories and protein compared to pizza. Whereas, a pizza would have more carbohydrates compared to a hamburger.

3. Explain which model validation technique you're using and why.

The model validation technique I used is K fold. This technique helps split the data and use performance metrics on test data to evaluate the model's performance. K fold cross validation is able to take all the performance metric and adds them together.

4. Explain why you did or did not choose to standardize your continuouos variables.

I chose to not standardize my continuous variable because the code would throw me an error. When I didn't standardize my continuous variable the code worked and gave me a good accuracy score.

5. Evaluate how the model performed. Explain.

My model performed well with the average accuracy score being 86.51%. Looking at all of the confusion matrix,

the off-diagonal it was able to predict most of it correctly.

At the end

1. Compare the performance of the 3 models using the accuracy, and the confusion matrix

All 3 models performed well with high accuracy scores. The lowest accuracy score was Model 3, 86.51%. While the highest accuracy score was Model 2, 91.94%. Model 3 is will be the one that will most to make an error compared to the other models. All 3 confusion matrices's off-diagonal was able to predict most of it correctly.

Part II

In [1]:
```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import AgglomerativeClustering


from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture

from sklearn.metrics import silhouette_score

import scipy.cluster.hierarchy as sch
from matplotlib import pyplot as plt

%matplotlib inline
```

In [2]:
```
ayenapapas = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392ParlettPelleriti/master/Data/KrispyKreme.csv")
ayenapapas.head()
```

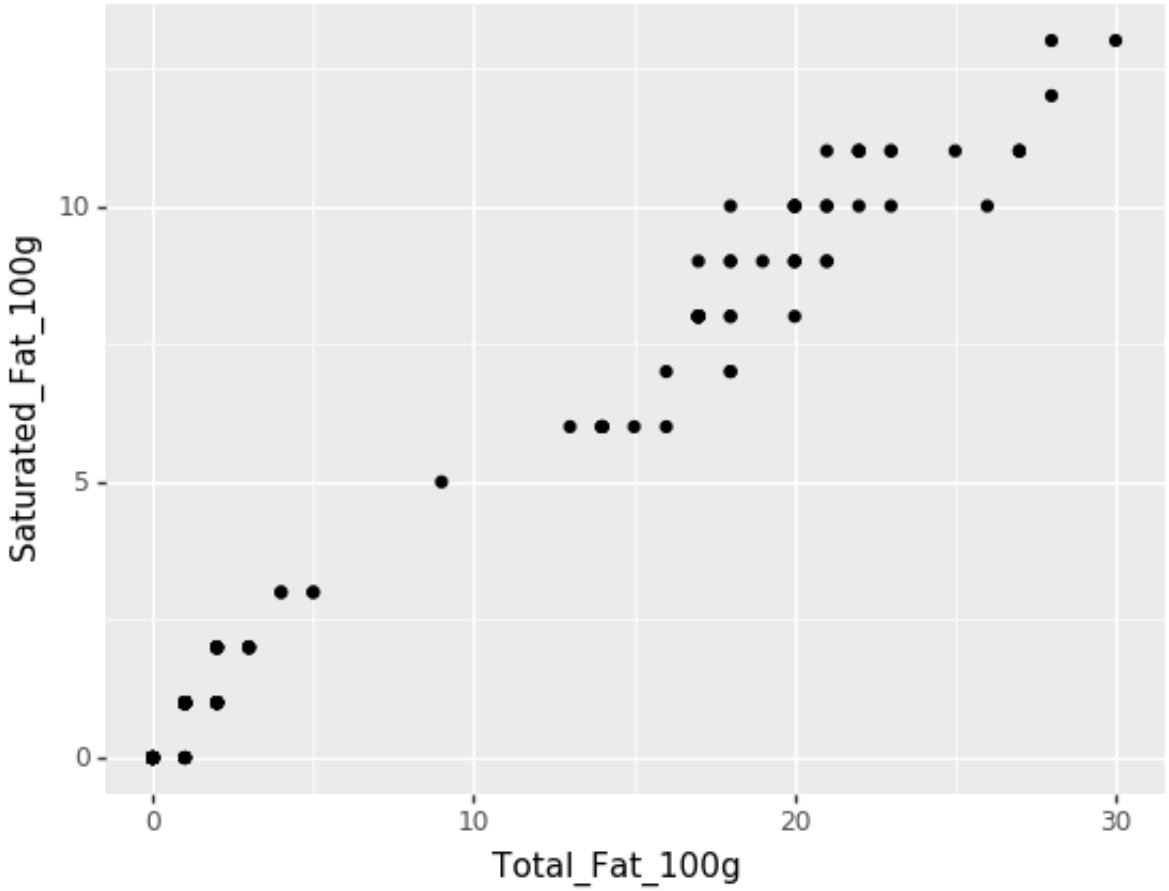
Out[2]:

	Restaurant_Item_Name	restaurant	Restaurant_ID	Item_Name	Item_Description	Food_Cate
0	Krispy Kreme Apple Fritter	Krispy Kreme	49	Apple Fritter	Apple Fritter, Doughnuts	Baked Go
1	Krispy Kreme Chocolate Iced Cake Doughnut	Krispy Kreme	49	Chocolate Iced Cake Doughnut	Chocolate Iced Cake Doughnut, Doughnuts	Baked Go
2	Krispy Kreme Chocolate Iced Custard Filled Dou...	Krispy Kreme	49	Chocolate Iced Custard Filled Doughnut	Chocolate Iced Custard Filled Doughnut, Doughnuts	Baked Go
3	Krispy Kreme Chocolate Iced Glazed Doughnut	Krispy Kreme	49	Chocolate Iced Glazed Doughnut	Chocolate Iced Glazed Doughnut, Doughnuts	Baked Go
4	Krispy Kreme Chocolate Iced Glazed Cruller Dou...	Krispy Kreme	49	Chocolate Iced Glazed Cruller Doughnut	Chocolate Iced Glazed Cruller Doughnut, Doughnuts	Baked Go

5 rows × 32 columns

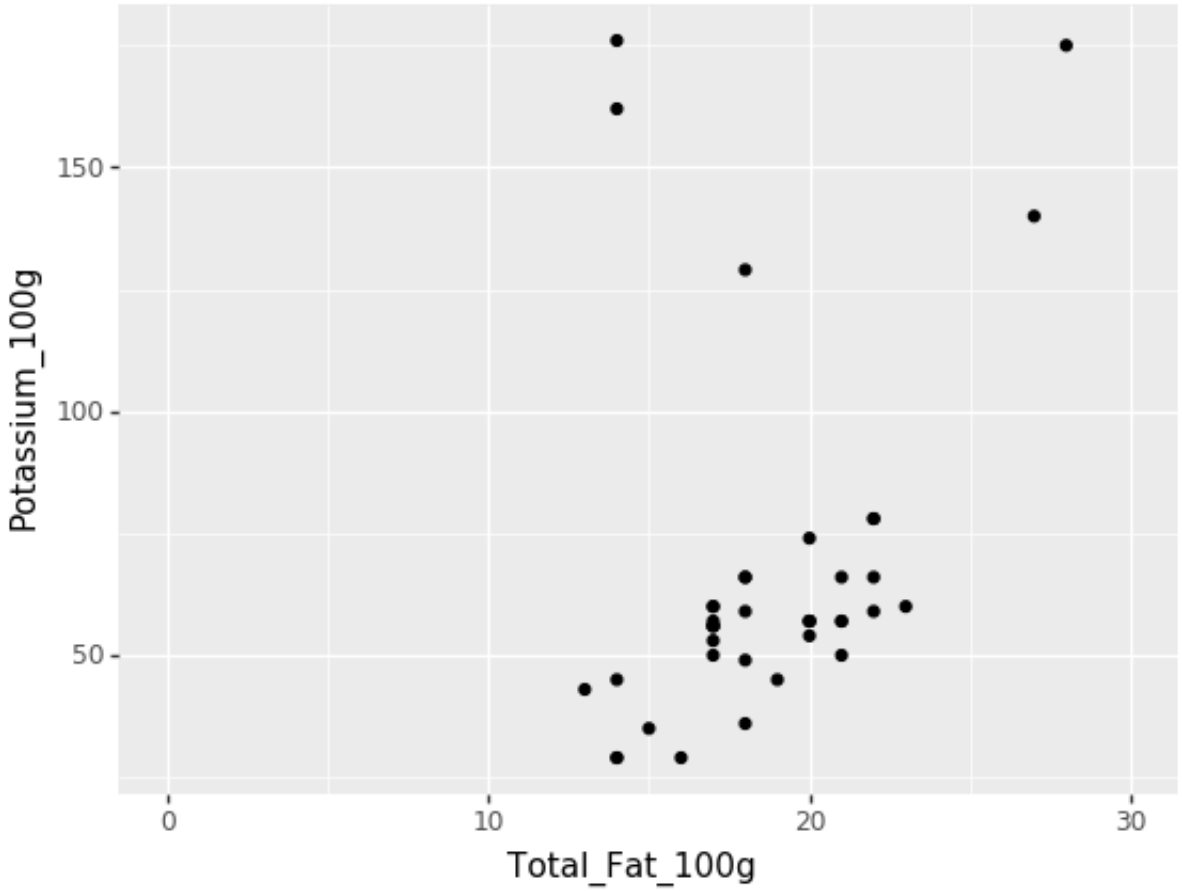
1. Explore Data (with ggplot)

In [7]:
```
(ggplot(ayenapapas, aes(x = "Total_Fat_100g", y = "Saturated_Fat_100g"
)) + geom_point())
```



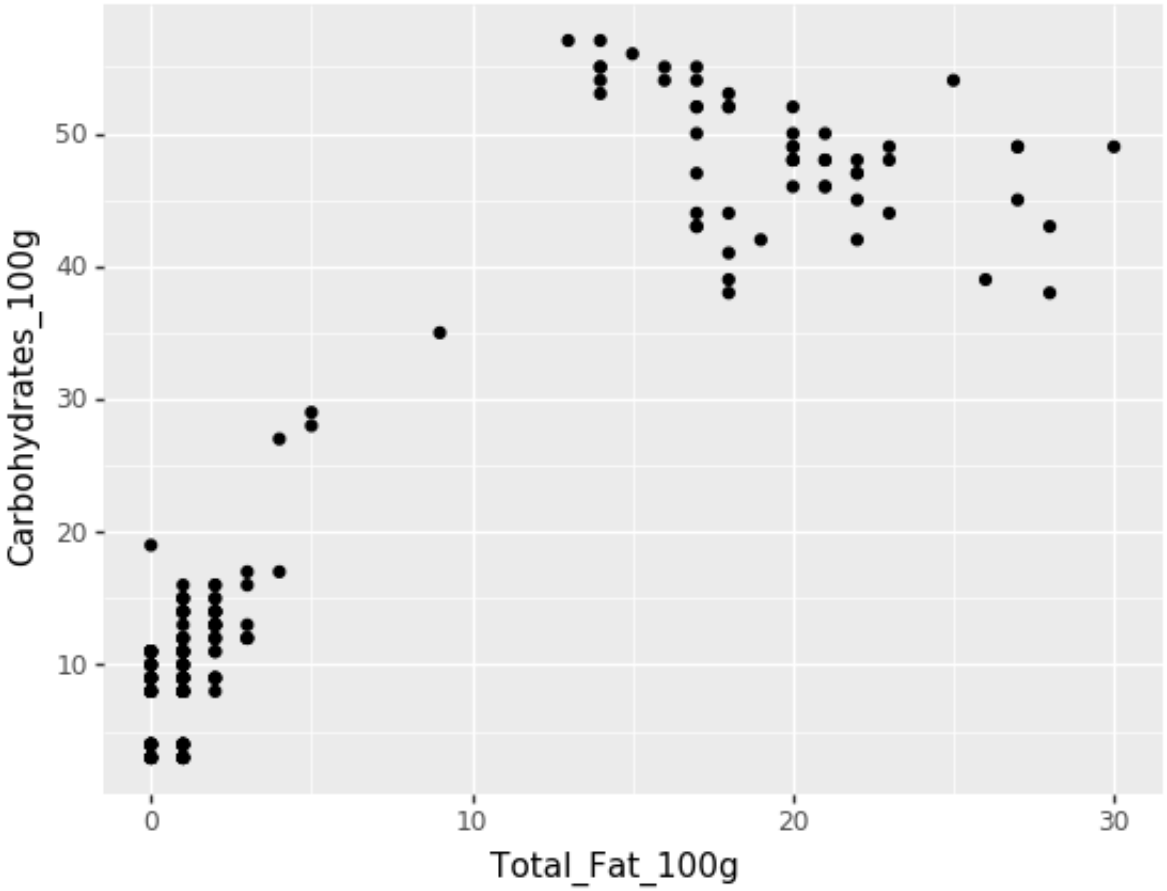
Out[7]: <ggplot: (304557621)>

In [34]:
```
(ggplot(ayenapapas, aes(x = "Total_Fat_100g", y = "Potassium_100g")) +
geom_point())
```



Out[34]: <ggplot: (306564201)>

In [35]:
```
(ggplot(ayenapapas, aes(x = "Total_Fat_100g", y = "Carbohydrates_100g"
)) + geom_point())
```



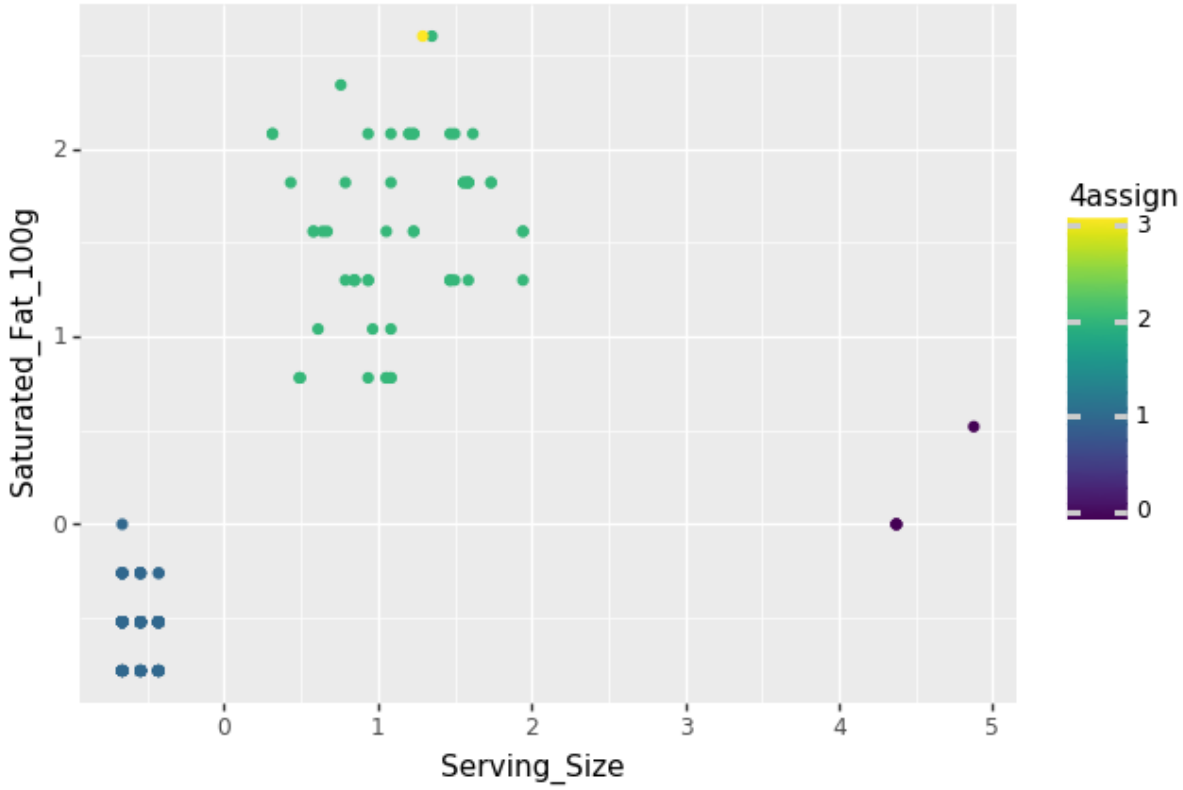
Out[35]: <ggplot: (305353501)>

K-Means Model

In [8]:
```
features = ["Serving_Size", "Total_Fat_100g", "Saturated_Fat_100g", "Trans_Fat_100g", "Carbohydrates_100g"]
X = ayenapapas[features]
Xdf = X

#standardize only the "total"
z = StandardScaler()
z.fit(X[features])
X[features] = z.fit_transform(X[features])

n_clusters = [2,3,4,5]
sil = []

for n in n_clusters:
    KM = KMeans(n_clusters = n)
    KM.fit(X)

    membership = KM.predict(X)
    sil.append(silhouette_score(X, membership))

    colName = str(n) + "assign"
    Xdf[colName] = membership

print(sil)
```

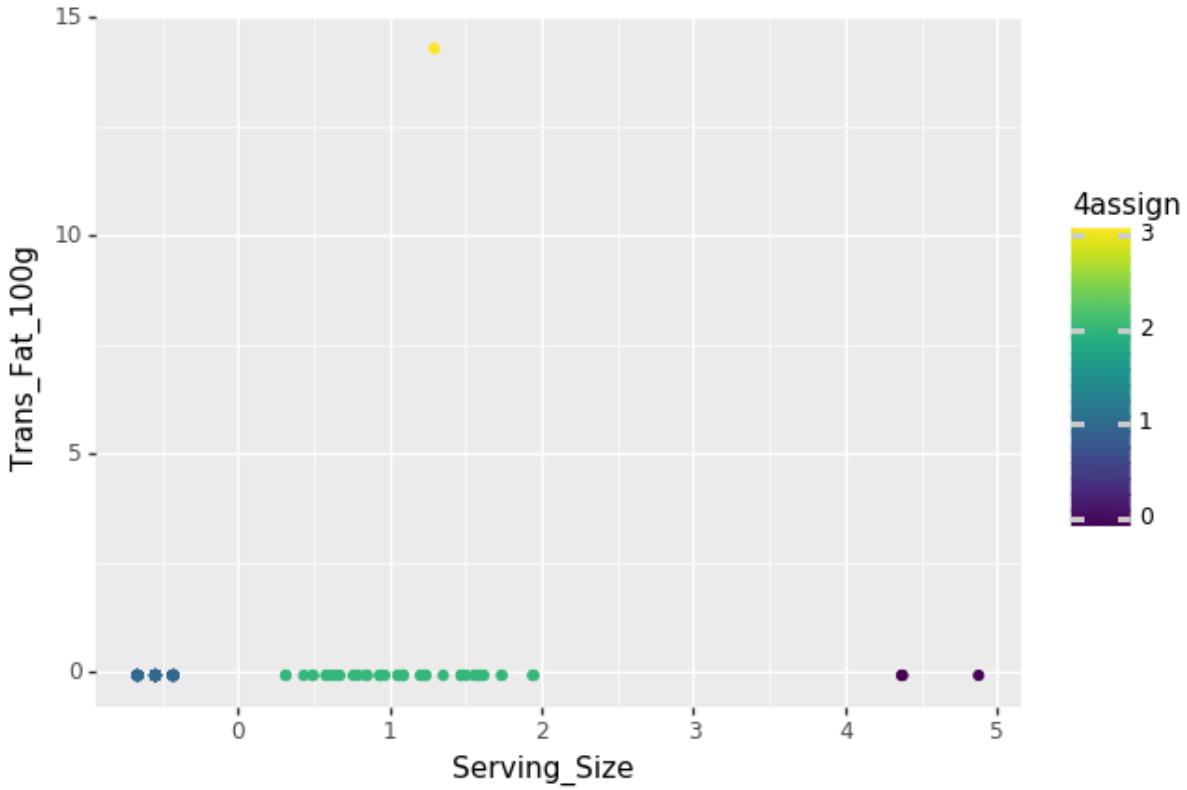
```
[0.8179886477427847, 0.8345835290228476, 0.8579736121542978, 0.7539259397471559]
```

In [10]:
```
(ggplot(Xdf, aes(x = "Serving_Size", y = "Saturated_Fat_100g", color = "4assign")) +geom_point())
```



Out[10]: <ggplot: (309496893)>

In [12]:
```
(ggplot(Xdf, aes(x = "Serving_Size", y = "Trans_Fat_100g", color = "4a
ssign")) + geom_point())
```



Out[12]: <ggplot: (310877585)>

2. Explain which variables you're using to predict the outcome.

The variables I am using to predict the outcome is total fat 100g, saturated fat 100g, trans fat 100g, and carbohydrates 100g. I am using these variables to predict the outcome because usually all of these variables are very unhealthy to the human body. Trans fat and saturated fat create LDL cholesterol that can lead to coronary artery disease. With too much carbohydrates can lead to high blood sugar levels that can result in more fat to build for the human body and possibly lead to diabetes and other health issues.

3. Evaluate how the model performed using silhoutte scores. Look at different numbers of clusters (like k = 3, 5). Which number of clusters is the best fit?

The fourth cluster is the best fit because it has the highest percentage, 85.80%, compared to the rest of the clusters.

4. Describe the clusters

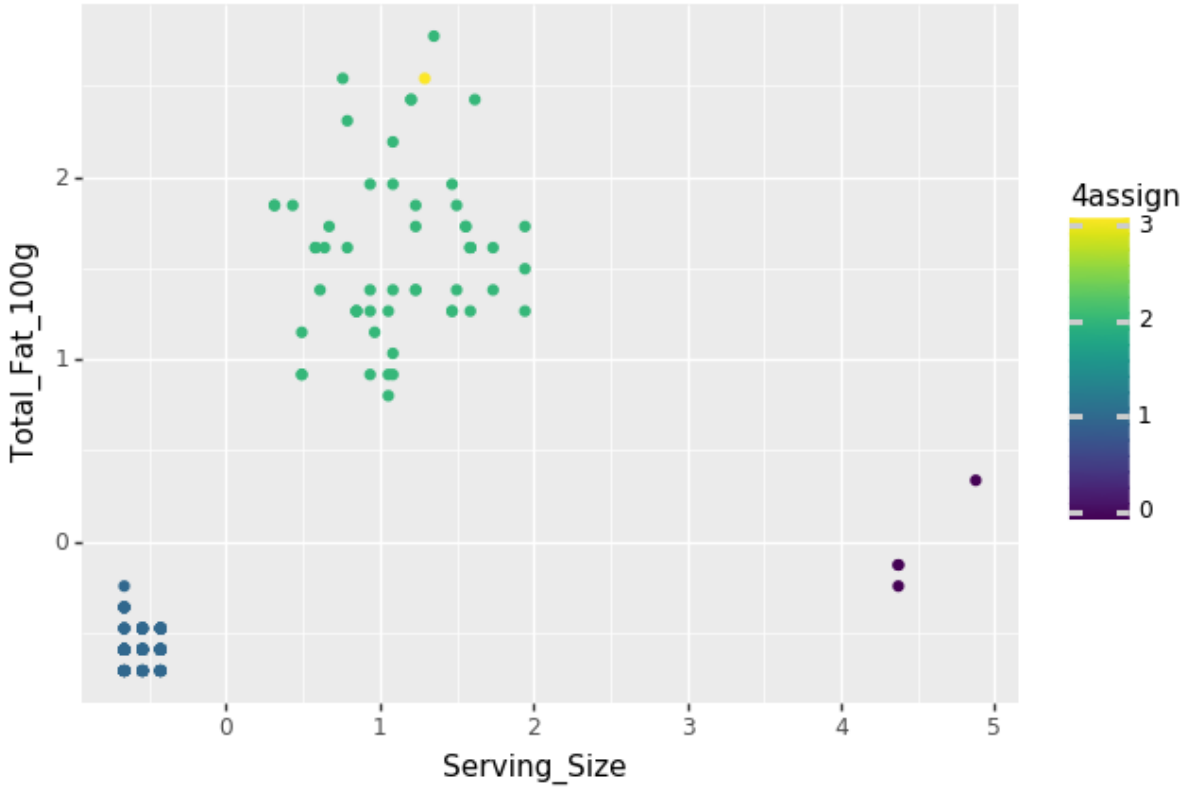
Looking at the graph saturated fat 100g, the blue colored points show that with no servings the value of the saturated fat would be 0g or below 0g. The green and yellow colored points is in the range for 0.5-2 servings. This group has the highest amount of saturated fat. Surprisingly, the purple points that have more than 4 servings have a lower saturated fat than serving size of 1 and 2 (green and yellow points). Looking at the graph for trans fat, most of the points (color blue, green, purple) have a trans fat of 0g. The yellow point is the only one to have a value higher than 0g. We can assume that this is an outlier.

Gaussian Mixture Model

In [36]:
```
features = ["Serving_Size", "Total_Fat_100g", "Saturated_Fat_100g", "T
rans_Fat_100g", "Carbohydrates_100g"]
X = ayenapapas[features]
Xdf = X

z = StandardScaler()
z.fit(X[features])
X[features] = z.fit_transform(X[features])
n_components = [2,3,4,5]
sil = []

for n in n_components:
    gmm = GaussianMixture(n_components = n)
    gmm.fit(X)

    clusters = gmm.predict(X)
    sil.append(silhouette_score(X, clusters))

    colName = str(n) + "assign"
    Xdf[colName] = clusters

print(sil)
```

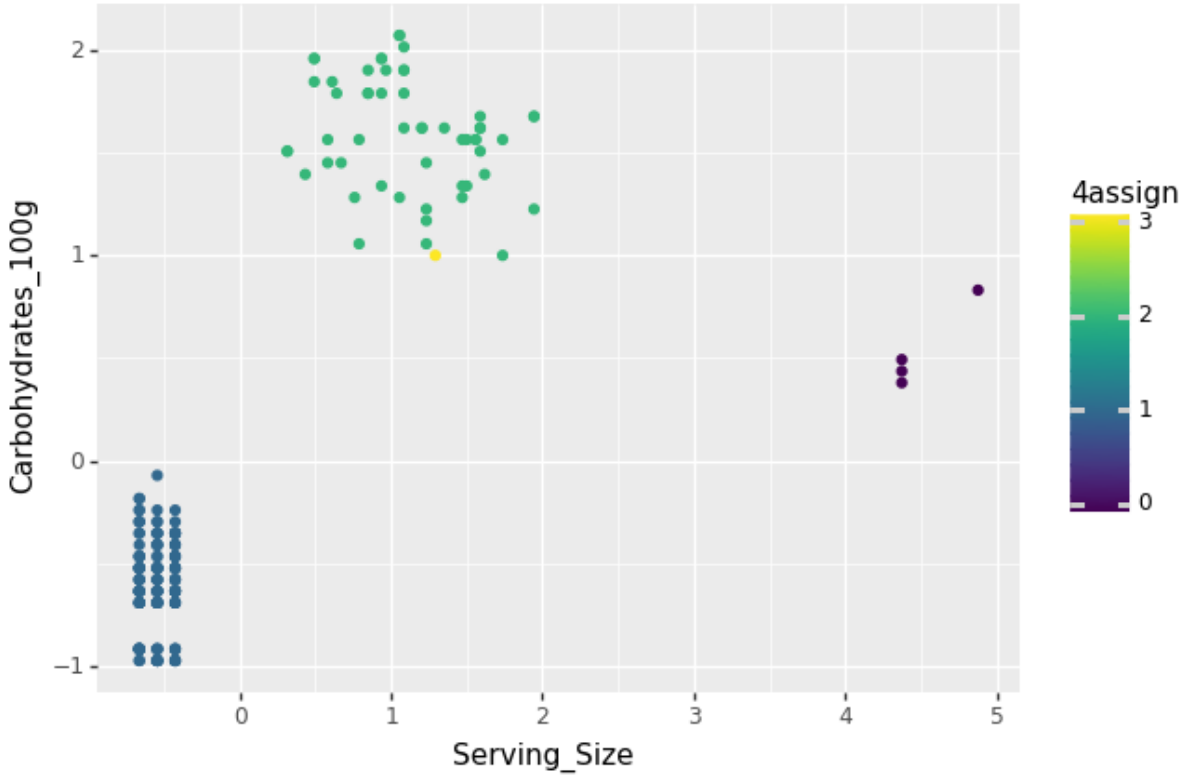
```
[0.8179886477427847, 0.8345835290228476, 0.8579736121542978, 0.76853
36362576782]
```

In [15]:
```
(ggplot(Xdf, aes(x = "Serving_Size", y = "Total_Fat_100g", color = "4a
ssign")) + geom_point())
```



Out[15]: <ggplot: (311303013)>

In [17]:
```
(ggplot(Xdf, aes(x = "Serving_Size", y = "Carbohydrates_100g", color = "4assign")) + geom_point())
```



Out[17]: <ggplot: (311303109)>

2. Explain which variables you're using to predict the outcome.

The variables I am using to predict the outcome is total fat 100g, saturated fat 100g, trans fat 100g, and carbohydrates 100g. I am using these variables to predict the outcome because usually all of these variables are very unhealthy to the human body. Trans fat and saturated fat create LDL cholesterol that can lead to coronary artery disease. With too much carbohydrates can lead to high blood sugar levels that can result in more fat to build for the human body and possibly lead to diabetes and other health issues.

3. Evaluate how the model performed using silhoutte scores. Look at different numbers of clusters (like k = 3, 5). Which number of clusters of clusters is the best fit?

The fourth cluster is the best fit because it has the highest percentage, 85.80%, compared to the rest of the

clusters.

4. Describe the clusters

Looking at the graph total fat, the blue colored points show that with no servings the value of the total fat would be 0g or below 0g. The green and yellow colored points is in the range for 0.5-2 servings. This group has the highest amount of total fat. Surprisingly, the purple points that have more than 4 servings have a lower total fat than serving size of 0.5-2 (green and yellow points). Looking at the graph carbohydrates, the blue colored points show that with no servings the value of the carbohydrates would be 0g or below 0g. The green and yellow colored points is in the range for 0.5-2 servings. This group has the highest amount of carbohydrates. Surprisingly, the purple points that have more than 4 servings have a lower total fat than serving size of 0.5-2 (green and yellow points).

At the end:

1. Compare the clusters obtained by the two models. Overall are they similar? or really different?

The clusters by the two models are very similar. Both models had the same high percentage of 85.80%. Both clusters also indicate that with a serving size or 0.5-2 would result in a lot of total fat, trans fat, saturated fat, and carbohydrates.