Project #2

May 1, 2020

[30]:
```
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.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder

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

from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import plot_confusion_matrix

from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelBinarizer

%matplotlib inline
```

1 Part 1

[31]:
```
bp = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/
↪CPSC392ParlettPelleriti/master/Data/burgersOrPizza.csv")
bp.head()
```

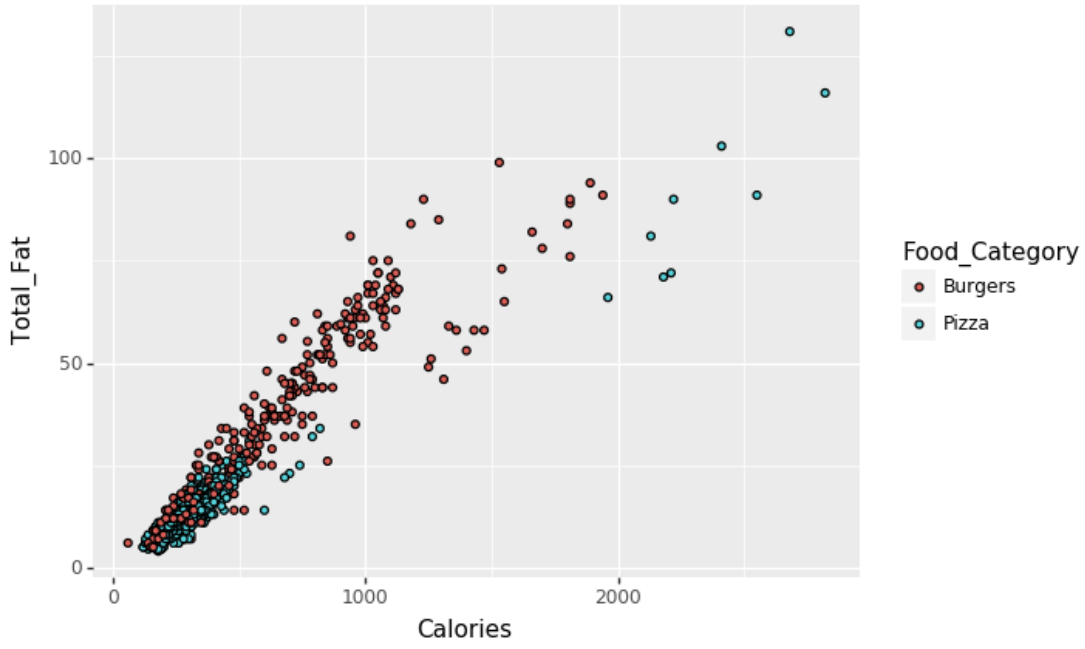
[31]:

```
                          Item_Name  \
0             Chicken n Cheese Slider
1          Corned Beef n Cheese Slider
2                   Ham n Cheese Slider
3  Jalapeno Roast Beef n Cheese Slider
4           Roast Beef n Cheese Slider

                                  Item_Description Food_Category  Calories  \
0  Chicken n Cheese Slider on Mini Bun w/ Chicken…       Burgers     290.0
1  Corned Beef n Cheese Slider on Mini Bun w/ Cor…       Burgers     220.0
2  Ham n Cheese Slider on Mini Bun w/ Roast Ham &…       Burgers     230.0
3  Jalapeno Roast Beef n Cheese Slider on Mini Bu…       Burgers     240.0
4  Roast Beef n Cheese Slider on Mini Bun w/ Roas…       Burgers     240.0

   Total_Fat  Saturated_Fat  Trans_Fat  Cholesterol  Sodium  Potassium  …  \
0       12.0            3.5        0.0         25.0   720.0        NaN  …
1        9.0            3.5        0.0         30.0   890.0        NaN  …
2        9.0            3.5        0.0         30.0   750.0        NaN  …
3       11.0            4.5        0.0         30.0   670.0        NaN  …
4       11.0            4.5        0.0         30.0   670.0        NaN  …

   Total_Fat_100g  Saturated_Fat_100g  Trans_Fat_100g  Cholesterol_100g  \
0              12                   4             0.0                25
1              10                   4             0.0                33
2              10                   4             0.0                33
3              11                   5             0.0                31
4              12                   5             0.0                33

   Sodium_100g  Potassium_100g  Carbohydrates_100g  Protein_100g  Sugar_100g  \
0          727             NaN                  30            15         1.0
1          978             NaN                  23            15         1.0
2          824             NaN                  24            14         3.0
3          684             NaN                  21            14         1.0
4          736             NaN                  23            15         1.0

   Dietary_Fiber_100g
0                 1.0
1                 1.0
2                 1.0
3                 1.0
4                 1.0
```

[5 rows x 25 columns]

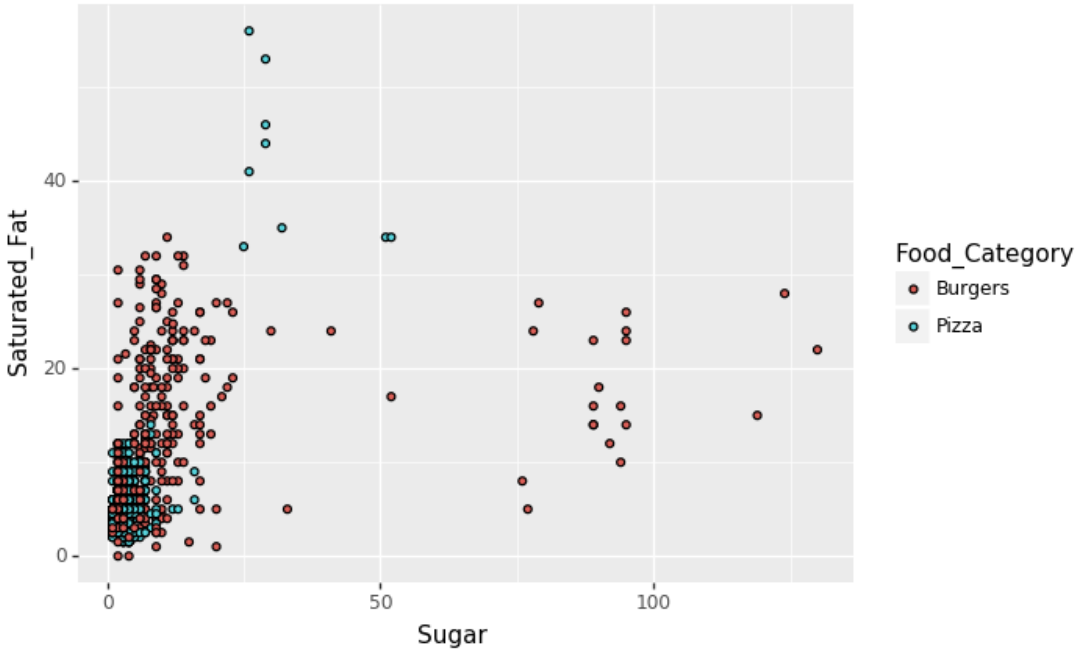
2 Explore Data

[39]:
```
ggplot(bp, aes("Calories", "Total_Fat")) + geom_point(aes(fill =␣
↪"Food_Category"))
```



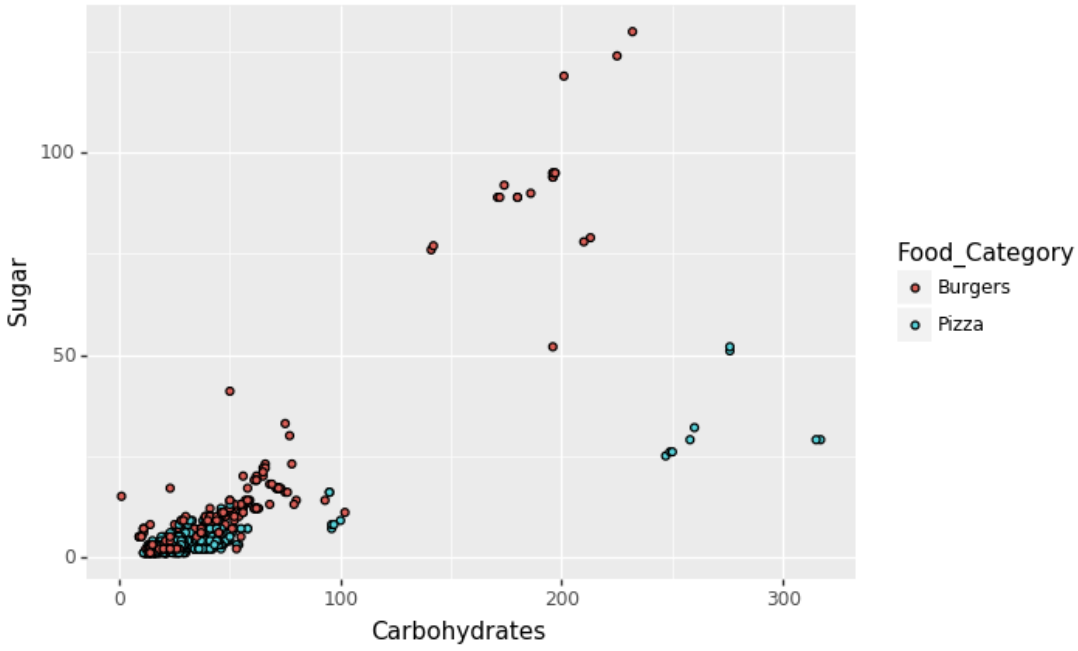
[39]: <ggplot: (289976429)>

[41]:
```
ggplot(bp, aes("Sugar", "Saturated_Fat")) + geom_point(aes(fill =␣
↪"Food_Category"))
```



[41]: <ggplot: (289880601)>

[43]:
```
ggplot(bp, aes("Carbohydrates", "Sugar", fill = "Food_Category")) + geom_point()
```



[43]: <ggplot: (302510417)>

3 Variable Selection, Model Selection, and Z-standarization

I am chosing to use all continuous variables to give my model maximum prediction power. However I do remove all the na's. Doing this will help my prediction without overwelming the model because it is not a large dataset.

Due to the fact that this is a small dataset, I chose to use k fold cross validation with 8 folds. Though running LOOCV would not burden the model because of the size of the dataset, I felt K fold would still be able to do the trick especially looking at 10 different folds.

I chose to standardize my variables because they're measured in different units and scales.

4 Loading in Data / Zscoring

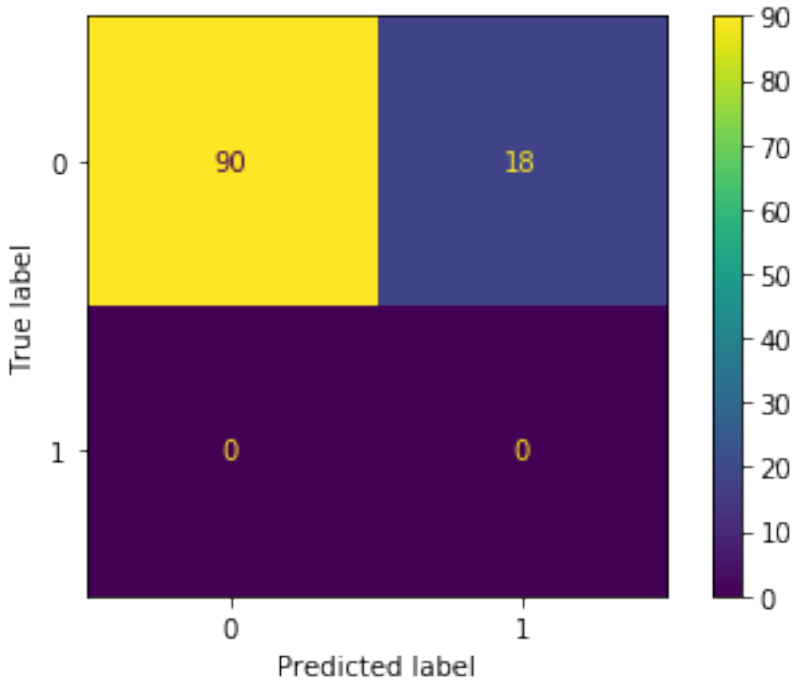
[45]:
```
features = bp.columns[1:20]
X = bp[features]
y = bp["Food_Category"]

bp = bp.dropna()

b = LabelBinarizer()
y = b.fit_transform(y)

z = StandardScaler()
z.fit(X)
X[features] = z.transform(X)
```

5 1. Logistic Regression

[46]:
```
kf = KFold(n_splits = 8)
kf.split(X)
lr = LogisticRegression()
acc = []

for train_indices, test_indices in kf.split(X):
    X_train = X.iloc[train_indices]
    X_test  = X.iloc[test_indices]
    y_train = y[train_indices]
    y_test  = y[test_indices]

    model = lr.fit(X_train, y_train)
    acc.append(accuracy_score(y_test, model.predict(X_test)))

chosen_k = max(acc)
```

```
lr_final = lr.fit(X_train,y_train)
plot_confusion_matrix(lr_final, X_test, y_test)
print(lr_final.score(X_test,y_test))
```

```
0.8333333333333334
```



This model appeared to perform very well with an accuracy score of 83.3% I used the kfold model to measure the performance on 10 different models. The best models had 90 correct predictions and 18 incorrect predictions.

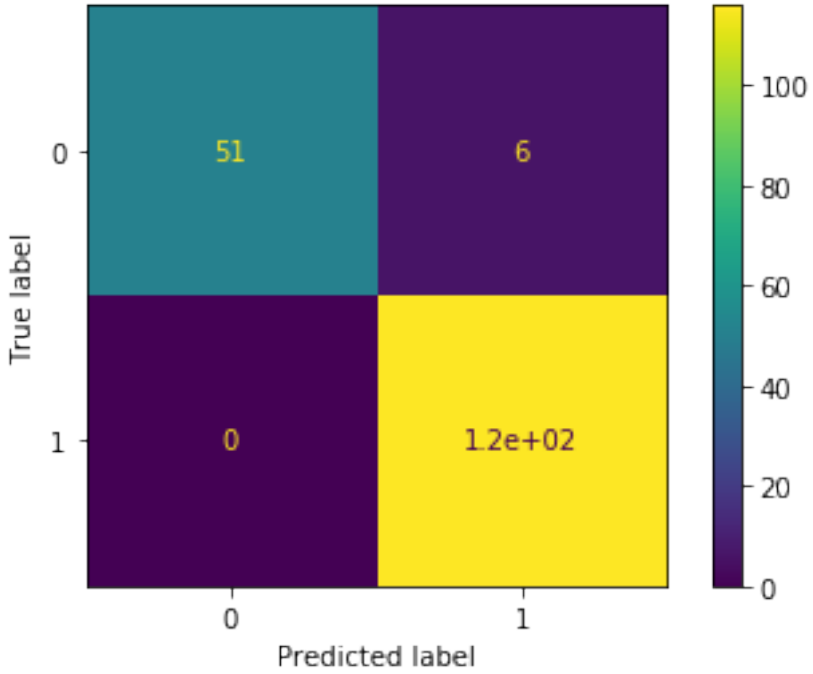
6 2. K Nearest Neighbors

[48]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

poss_k = [3,4,5,6,7]
acc = {}

for k in poss_k:
    kf = KFold(n_splits = 5)
    knn = KNeighborsClassifier(n_neighbors = k)
    acc[k] = np.mean(cross_val_score(knn, X_train, y_train, cv = kf))

print(acc)
```

```
chosen_k = max(acc, key=acc.get)

knn_final = KNeighborsClassifier(n_neighbors = chosen_k)
knn_final.fit(X_train,y_train)
plot_confusion_matrix(knn_final, X_test, y_test)

knn_final.score(X_test,y_test)
```

```
{3: 0.9595349807110832, 4: 0.9653008028359921, 5: 0.9551871546241267, 6:
0.9522990303409447, 7: 0.9494213324992179}
```

[48]:
```
0.9653179190751445
```



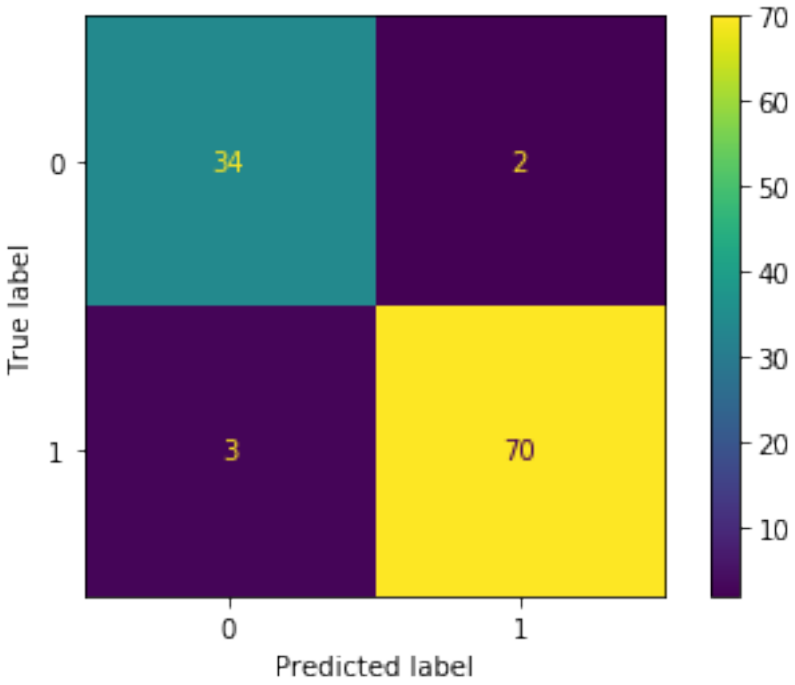
My K Nearest Neighbors model performed alot better than the Logistics Regression model with my best model performing at an accuracy score of 96.5% Predicting 171 correct predictions and 6 incorrect predictions

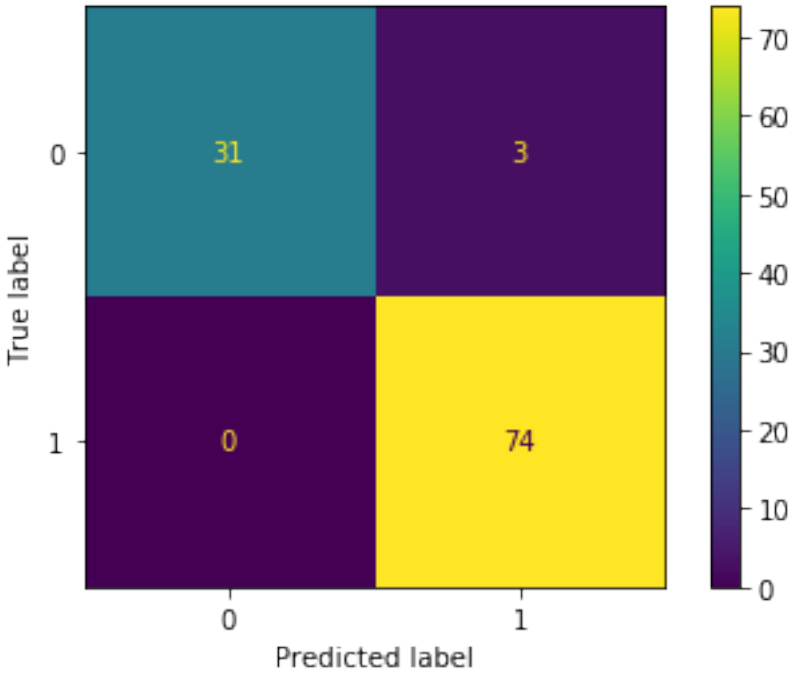
7 3. Decision Tree

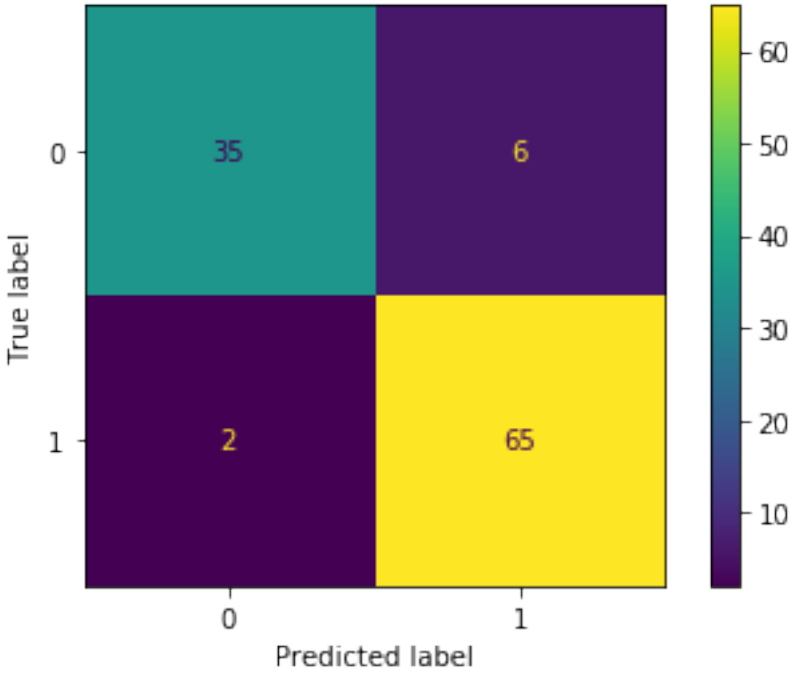
[49]:
```
kf = KFold(8, shuffle = True)
acc = []
depth = []

for train, test in kf.split(X):
```

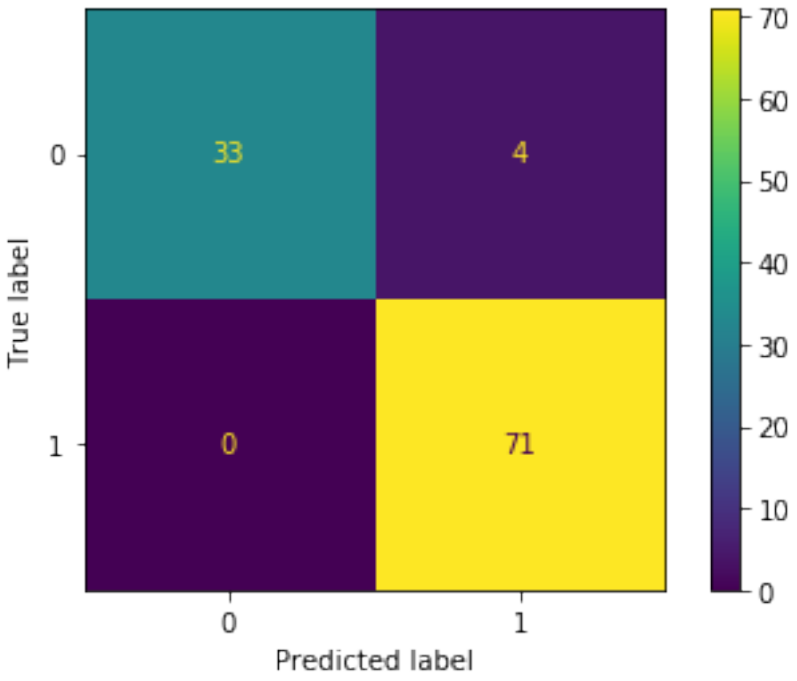
```
    X_train = X.iloc[train,]
    X_test = X.iloc[test,]
    y_train = y[train]
    y_test = y[test]

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

    acc.append(tree.score(X_test,y_test))
    depth.append(tree.get_depth())
    plot_confusion_matrix(tree_mod, X_test, y_test)

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

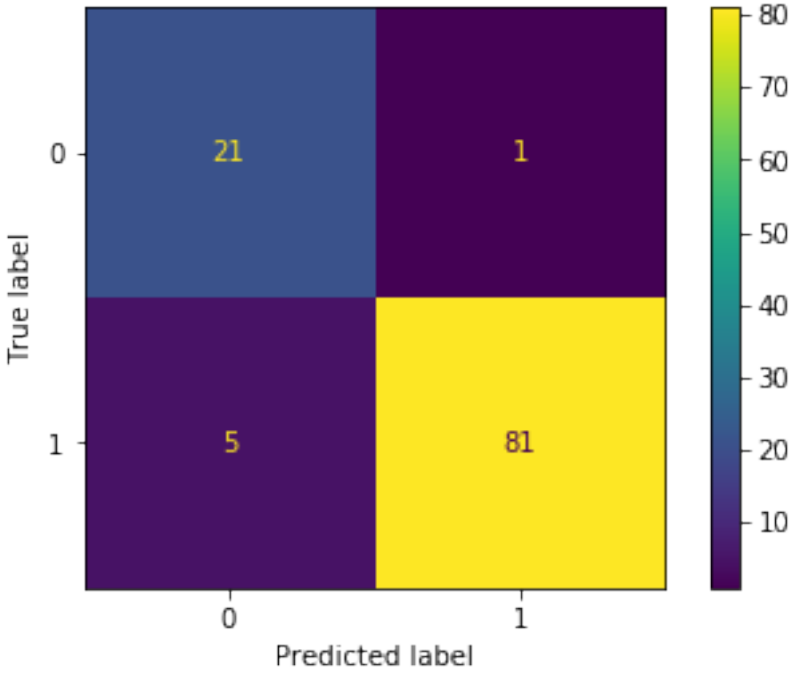
```
[0.9541284403669725, 0.9722222222222222, 0.9259259259259259, 0.9629629629629629,
0.9444444444444444, 0.9722222222222222, 0.9537037037037037, 0.9814814814814815]
0.958386425416242
[15, 13, 13, 16, 15, 13, 12, 15]
```

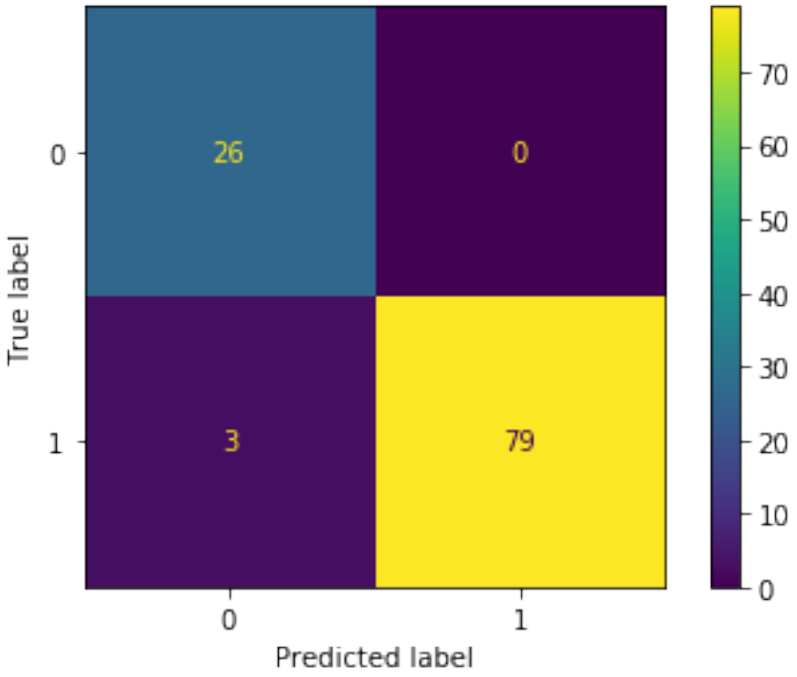


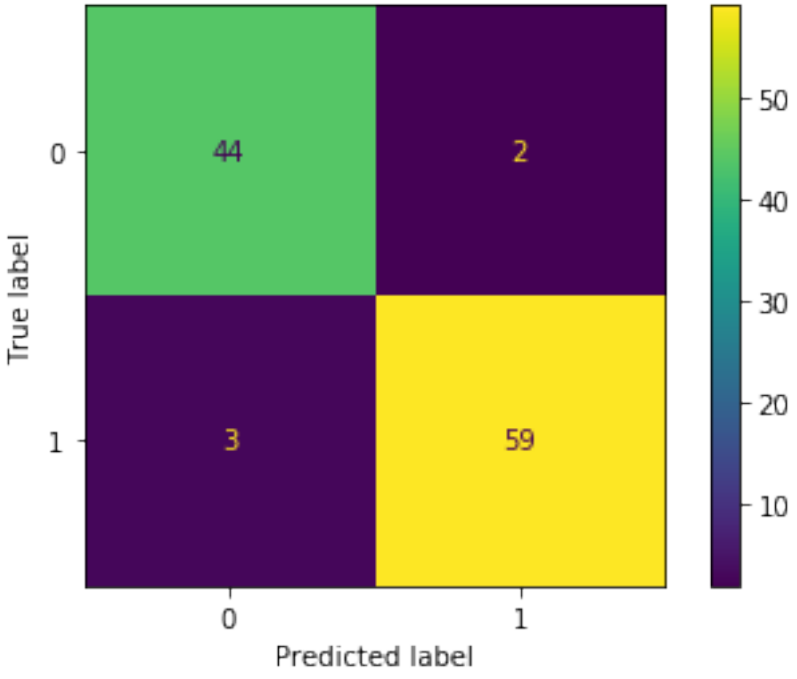

31
3
0
74
0
1
True label
0
1
Predicted label
70
60
50
40
30
20
10
0

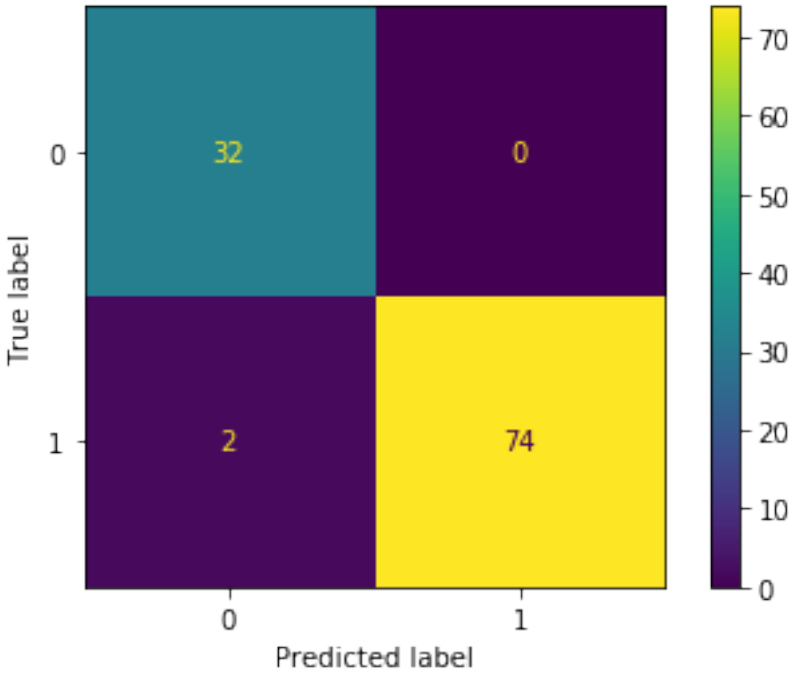

35
6
2
65
0
1
True label
0
1
Predicted label
60
50
40
30
20
10


33
4
0
71
0
1
True label
0
1
Predicted label
70
60
50
40
30
20
10
0


21
1
5
81
0
1
True label
0
1
Predicted label
80
70
60
50
40
30
20
10


26
0
3
79
0
1
True label
Predicted label
70
60
50
40
30
20
10
0


44
2
3
59
0
1
True label
Predicted label
50
40
30
20
10



My Decision Tree model appeared to perform very well also with an average accuracy score of 95.8% across 8 folds.

8 Best Model

Logistic = 83% accuracy score

KNN = 96% accuracy score

Decision Tree = 95% accuracy score

According to these accuracy scores KNN and Decision Trees appear to be the best models for prediction. Some of the Logistic Regression models inaccuracy could be because the model may be overfit which is reflected in the accuracy score.

9 Part II

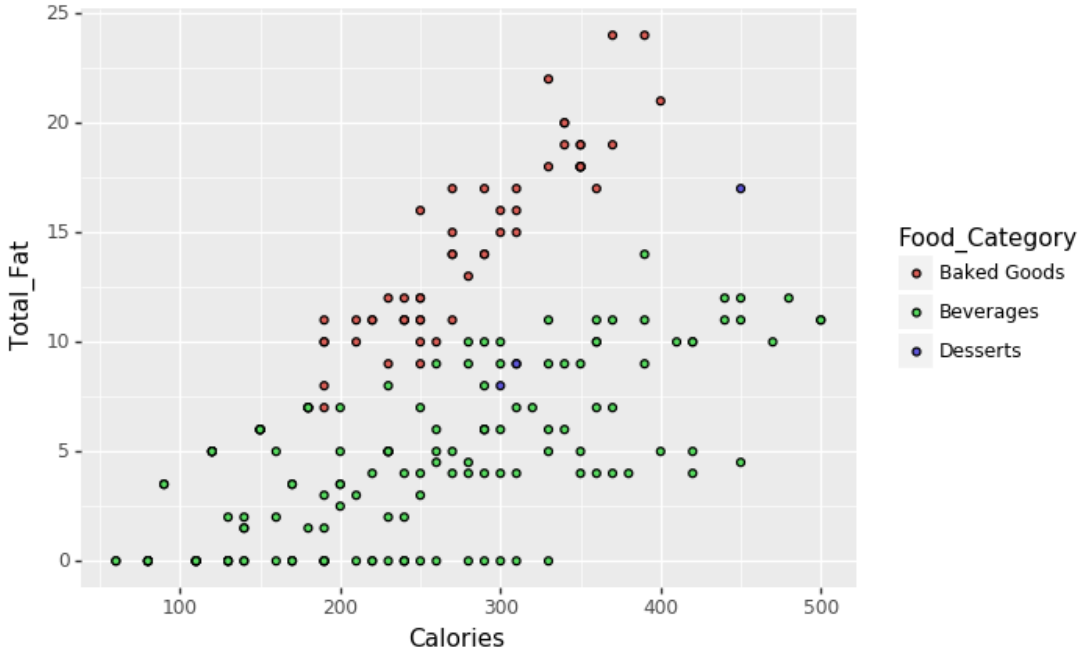
[50]:
```
kk = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/
↪CPSC392ParlettPelleriti/master/Data/KrispyKreme.csv")
kk.head()
```

[50]:
```
                                Restaurant_Item_Name    restaurant  \
0                      Krispy Kreme Apple Fritter  Krispy Kreme
1          Krispy Kreme Chocolate Iced Cake Doughnut  Krispy Kreme
2  Krispy Kreme Chocolate Iced Custard Filled Dou…  Krispy Kreme
```

```
3       Krispy Kreme Chocolate Iced Glazed Doughnut  Krispy Kreme
4  Krispy Kreme Chocolate Iced Glazed Cruller Dou…  Krispy Kreme

   Restaurant_ID                                Item_Name  \
0             49                           Apple Fritter
1             49            Chocolate Iced Cake Doughnut
2             49  Chocolate Iced Custard Filled Doughnut
3             49          Chocolate Iced Glazed Doughnut
4             49  Chocolate Iced Glazed Cruller Doughnut

                                  Item_Description Food_Category  \
0                          Apple Fritter, Doughnuts   Baked Goods
1            Chocolate Iced Cake Doughnut, Doughnuts   Baked Goods
2  Chocolate Iced Custard Filled Doughnut, Doughnuts   Baked Goods
3          Chocolate Iced Glazed Doughnut, Doughnuts   Baked Goods
4  Chocolate Iced Glazed Cruller Doughnut, Doughnuts   Baked Goods

   Serving_Size  Serving_Size_text Serving_Size_Unit Serving_Size_household  \
0           100                NaN                g                     NaN
1            71                NaN                g                     NaN
2            85                NaN                g                     NaN
3            63                NaN                g                     NaN
4            70                NaN                g                     NaN

   …  Total_Fat_100g  Saturated_Fat_100g  Trans_Fat_100g  Cholesterol_100g  \
0  …              19                   9               0                 0
1  …              18                   7               0                35
2  …              18                   8               0                 0
3  …              17                   8               0                 0
4  …              14                   6               0                29

   Sodium_100g  Potassium_100g  Carbohydrates_100g  Protein_100g  Sugar_100g  \
0          110            45.0                  42             4          26
1          437            49.0                  52             4          27
2          165            59.0                  44             5          20
3          143            56.0                  52             5          32
4          386            29.0                  57             4          37

   Dietary_Fiber_100g
0                 1.0
1                 NaN
2                 1.0
3                 NaN
4                 NaN

[5 rows x 32 columns]
```

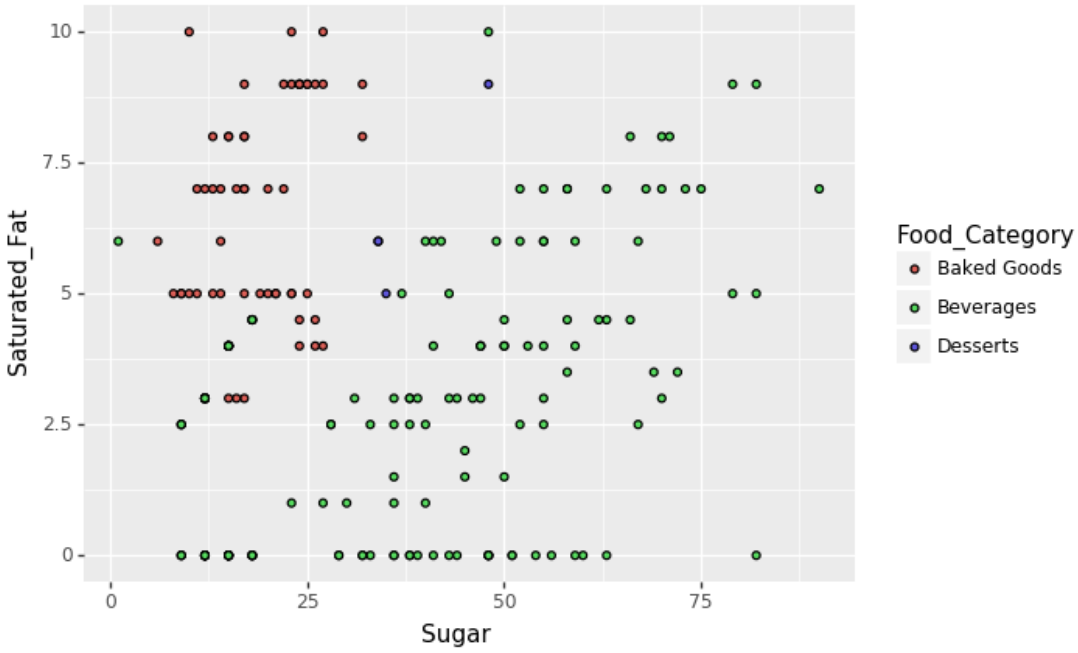
10 1. Explore Data (K-Means)

[51]:
```
ggplot(kk, aes("Calories", "Total_Fat")) + geom_point(aes(fill =␣
↪"Food_Category"))
```



[51]:
```
<ggplot: (304478981)>
```

[52]:
```
ggplot(kk, aes("Sugar", "Saturated_Fat")) + geom_point(aes(fill =␣
↪"Food_Category"))
```



[52]: <ggplot: (304373889)>

11 2. Variable Selection (K-Means)

I am going to use the variables: Calories, Total_Fat, Saturated_Fat, and Sugar. These variables appear to be the best when it comes to identifying which food Category the item of food is in.

12 3. Evaluate Model (K-Means)

[53]:
```
features = ["Calories","Total_Fat", "Saturated_Fat", "Sugar"]
X = kk[features]

z = StandardScaler()
X[features] = z.fit_transform(X)

n_components = [2,3,4,5,6]

sils = []
for n in n_components:
    km = KMeans(n_clusters = n)
    km.fit(X)
    membership = km.predict(X)
    X["cluster"] = membership
    sils.append(silhouette_score(X, membership))
```

```
print(sils)
```

```
[0.42234233743207067, 0.49575674269334263, 0.5124874255637523,
0.5932819918512732, 0.5299262302970517]
```

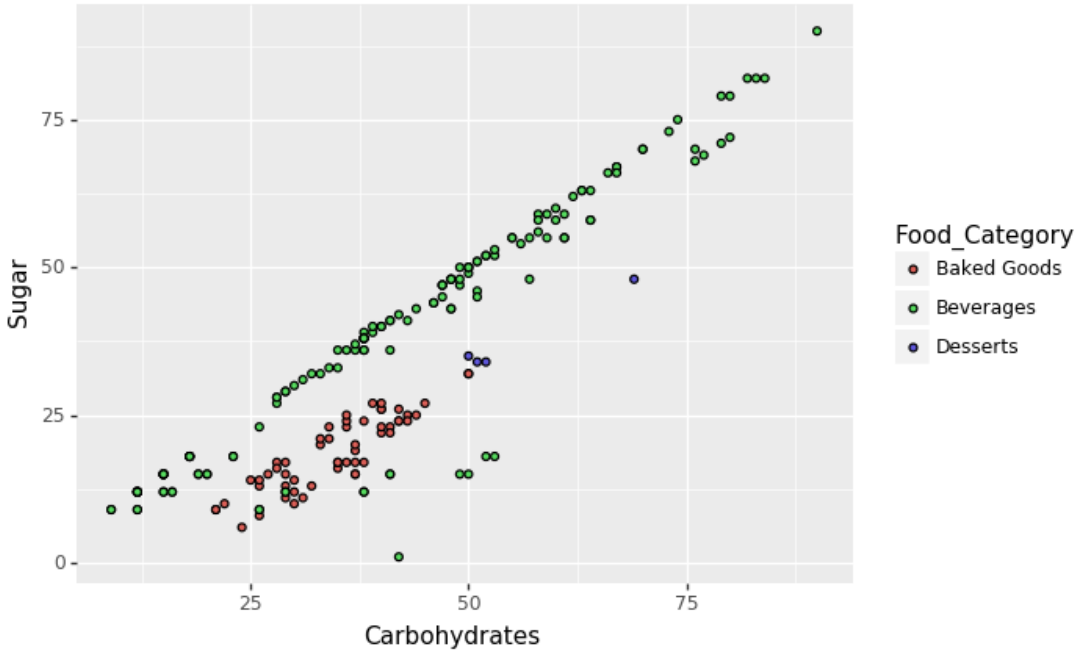
6 clusters was the best fit for the KNN model

13 4. Describe the cluster (K-Means)

The Clusters for the KNN model seem to be evenly dispersed. However on of the clusters seems to follow a positive linear path whereas the other cluster is more spread out and random.

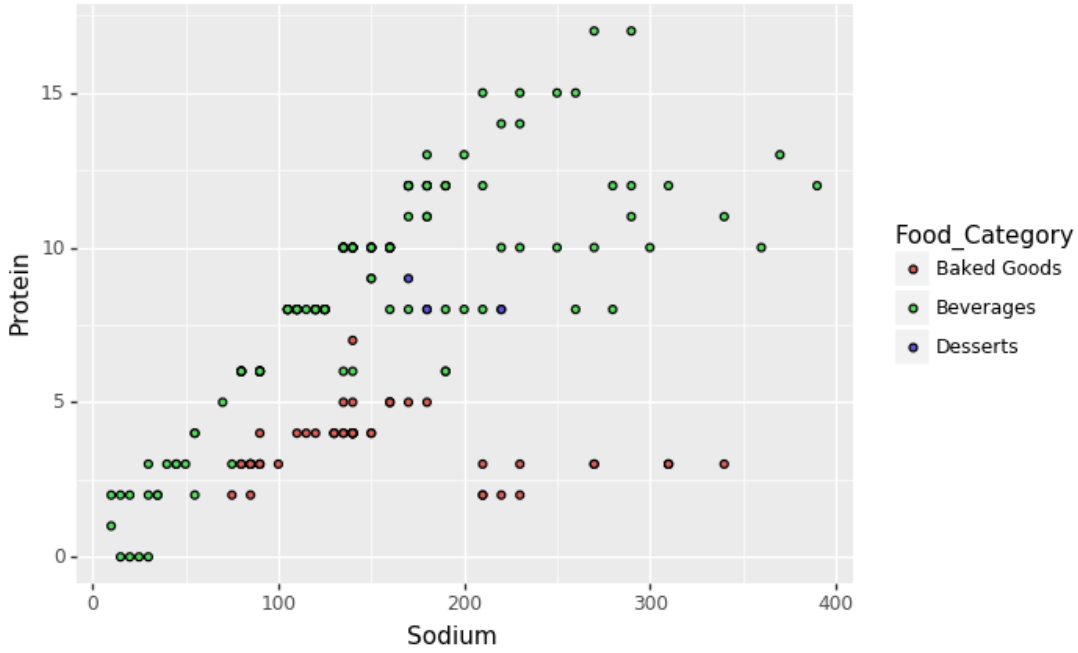
14 1. Explore Data (Gaussian)

[54]:
```
ggplot(kk, aes("Carbohydrates", "Sugar", fill = "Food_Category")) + geom_point()
```



[54]:
```
<ggplot: (302428997)>
```

[55]:
```
ggplot(kk, aes("Sodium", "Protein", fill = "Food_Category")) + geom_point()
```



[55]: <ggplot: (302429301)>

[56]:
```
features = ["Sugar", "Carbohydrates","Sodium", "Protein"]

X = kk[features]
z = StandardScaler()
X[features] = z.fit_transform(X)
Xdf = X


n_components = [2,3,4,5,6]

sils = []
for n in n_components:
    gmm = GaussianMixture(n_components = n)
    gmm.fit(X)
    colName = str(n) + "assign"
    clusters = gmm.predict(X)

    Xdf[colName] = clusters

    sils.append(silhouette_score(X, clusters))

print(sils)
```

```
[0.2707908552735736, 0.3468966626050098, 0.4549296028017138, 0.5587478131489594,
```

0.644772839490956]

6 clusters was the best fit for the Caussian Model

15 4. Describe the clusters

These clusters both seem to not be as random and spread out.

16 Compare the clusters obtained by the two models.

I think the clusters for each model are very similar and contain alot of the same members