final_project

March 25, 2023

1 Prediction of Fat Levels in Canadian Cheese

Author : Muntakim Rahman UBC Student Number : 71065221

2 Introduction

This **Jupyter Notebook** will be implementing *Machine Learning (ML)* models to predict the fat levels in *Canadian* cheeses.

2.1 Intended Outcome

We can utilize a rich understanding of the factors which yield high fat levels to reduce the risk of mass manufacturing an unsuccessful product.

We are already confident that the *Total Addressable Market (TAM)* is particularly concerned with fat level in cheeses (*as per a previous study on consumer tastes*). We would like to manufacture `lower fat` cheese products and this will act as our positive label.

2.2 What is Machine Learning?

Machine Learning is defined by *IBM* as *the use of statistical methods, algorithms that are trained to make classifications or predictions, and to uncover key insights in data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics.*

2.3 Cheese Classification

We will be identifying different features within our dataset in order to classify our cheese products as either `lower fat` or `higher fat`.

2.3.1 Estimators

We will be training, and evaluating a set of *ML* models (i.e. estimators), which will be compared to a baseline model, in our case a *Dummy Classifier*.

This will involve :

Tactful preprocessing of our data with imputation, scaling, feature transformations

Training the classifier models with an appropriately split and balanced dataset.

Assessing the model with evaluation metrics.

Picking the estimator with the best validation score and finetuning the hyperparameters.

2.4 Dataset Description

This dataset provides an overview of the different types of Canadian cheeses. The original data was found on the *Government of Canada's Open Government Portal* but has unfortunately been taken down. What we have here is a wrangled and partially cleaned, modified version of the original dataset.

3 Exploratory Data Analysis

3.1 Import Packages

[1]:
```
import numpy as np
import pandas as pd

import altair as alt

import datetime as dt

from sklearn.model_selection import train_test_split, cross_validate

from sklearn.impute import SimpleImputer
from sklearn.feature_extraction.text import CountVectorizer

from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import (
    OneHotEncoder,
    StandardScaler,
)

from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

import sklearn.metrics
from sklearn.metrics import make_scorer, accuracy_score, precision_score,␣
 ↪recall_score, f1_score
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix,␣
 ↪classification_report
```

```
from sklearn import set_config

alt.data_transformers.disable_max_rows()

from cheese_functions import *
```

[2]:
```
directory_df = pd.read_csv("data/canadianCheeseDirectory.csv")
display(directory_df.head())
```

```
   CheeseId                         CheeseNameEn  \
0       228                                  NaN
1       242                                  NaN
2       301  Provolone Sette Fette (Tre-Stelle)
3       303                                  NaN
4       319                                  NaN

                       CheeseNameFr       ManufacturerNameEn  \
0             Sieur de Duplessis (Le)                      NaN
1                 Tomme Le Champ Doré                      NaN
2  Provolone Sette Fette (Tre-Stelle)  Tre Stelle (Arla Foods)
3                      Geai Bleu (Le)                      NaN
4                          Gamin (Le)                      NaN

         ManufacturerNameFr ManufacturerProvCode ManufacturingTypeEn  \
0  Fromages la faim de loup                   NB           Farmstead
1  Fromages la faim de loup                   NB           Farmstead
2                       NaN                   ON          Industrial
3  Fromages la faim de loup                   NB           Farmstead
4  Fromages la faim de loup                   NB           Farmstead

  ManufacturingTypeFr                          WebSiteEn  \
0           Fermière                                NaN
1           Fermière                                NaN
2       Industrielle  http://www.trestelle.ca/english/
3           Fermière                                NaN
4           Fermière                                NaN

                          WebSiteFr  …  Organic    CategoryTypeEn  \
0                               NaN  …        0       Firm Cheese
1                               NaN  …        0  Semi-soft Cheese
2  http://www.trestelle.ca/francais/  …        0       Firm Cheese
3                               NaN  …        0    Veined Cheeses
4                               NaN  …        1  Semi-soft Cheese

    CategoryTypeFr MilkTypeEn MilkTypeFr MilkTreatmentTypeEn  \
0       Pâte ferme        Ewe     Brebis            Raw Milk
1  Pâte demi-ferme        Cow      Vache            Raw Milk
```

```
2       Pâte ferme        Cow      Vache         Pasteurized
3    Pâte persillée       Cow      Vache            Raw Milk
4  Pâte demi-ferme        Cow      Vache            Raw Milk

  MilkTreatmentTypeFr   RindTypeEn    RindTypeFr LastUpdateDate
0           Lait cru  Washed Rind  Croûte lavée     2016-02-03
1           Lait cru  Washed Rind  Croûte lavée     2016-02-03
2         Pasteurisé          NaN           NaN     2016-02-03
3           Lait cru          NaN           NaN     2016-02-03
4           Lait cru  Washed Rind  Croûte lavée     2016-02-03

[5 rows x 30 columns]
```

[3]:
```
data_df = pd.read_csv("data/cheese_data.csv")
display(data_df.head())
```

```
   CheeseId ManufacturerProvCode ManufacturingTypeEn  MoisturePercent  \
0       228                   NB           Farmstead             47.0
1       242                   NB           Farmstead             47.9
2       301                   ON          Industrial             54.0
3       303                   NB           Farmstead             47.0
4       319                   NB           Farmstead             49.4

                                      FlavourEn  \
0                                 Sharp, lactic
1               Sharp, lactic, lightly caramelized
2                          Mild, tangy, and fruity
3  Sharp with fruity notes and a hint of wild honey
4                                  Softer taste

                                 CharacteristicsEn  Organic  \
0                                         Uncooked        0
1                                         Uncooked        0
2  Pressed and cooked cheese, pasta filata, inter…        0
3                                              NaN        0
4                                              NaN        1

     CategoryTypeEn MilkTypeEn MilkTreatmentTypeEn   RindTypeEn  \
0       Firm Cheese        Ewe            Raw Milk  Washed Rind
1  Semi-soft Cheese        Cow            Raw Milk  Washed Rind
2       Firm Cheese        Cow         Pasteurized          NaN
3    Veined Cheeses        Cow            Raw Milk          NaN
4  Semi-soft Cheese        Cow            Raw Milk  Washed Rind

                         CheeseName   FatLevel
0            Sieur de Duplessis (Le)  lower fat
1                Tomme Le Champ Doré  lower fat
2  Provolone Sette Fette (Tre-Stelle)  lower fat
```

```
3                      Geai Bleu (Le)  lower fat
4                          Gamin (Le)  lower fat
```

3.2 Split Training and Test Datasets

Let's start by separating our training and test datasets. We're going to be working with a *80%* training and *20%* test set split and set our `random_state` variable to *77*.

3.2.1 Golden Rule of Machine Learning

We don't want the test data to influence our model **in any way**. This must act as completely unseen data during the model training and validation process.

[4]:
```
train_df, test_df = train_test_split(data_df, test_size = 0.2, random_state =␣
↪77)
```

[5]:
```
X_train, y_train = train_df.drop(columns = ['FatLevel']), train_df['FatLevel']
X_test, y_test = test_df.drop(columns = ['FatLevel']), test_df['FatLevel']
```

3.3 Exploratory Data Analysis

3.3.1 Observe Outputs

Let's start by plotting a bar chart showing the quantity of each `FatLevel` in the training data.

[6]:
```
fig_number = 1

fat_prop = alt.Chart(
    train_df,
    title = alt.TitleParams(
        text = f'Figure {fig_number} : Fat Levels for Canadian Cheeses',
        subtitle = '''Data found on the Government of Canada's Open Government␣
↪Portal''',
        fontSize = 20, subtitleFontSize = 15,
        anchor = 'start'
    )
).transform_joinaggregate(
    total = 'count(*)'
).transform_calculate(
    pct = '1 / datum.total'
).mark_bar().encode(
    x = alt.X('count()', title = 'Quantity'),
    y = alt.Y('FatLevel:N', title = 'Fat Level'),
    tooltip = [alt.Tooltip('sum(pct):Q', format = '.2%', formatType = 'number',␣
↪title = '% of Total')],
    color = alt.Color(
        'FatLevel:N',
        scale = alt.Scale(
            domain = ['lower fat', 'higher fat'],
```

```
            range = ['teal', 'crimson']
        ),
        legend = alt.Legend(title = 'Fat Level')
    )
).properties(
    width = 500, height = 300,
).configure_axis(labelFontSize = 12, titleFontSize = 15)

display(fat_prop)

fig_number += 1
```

```
alt.Chart(…)
```

Imbalanced Data There's a high percentage of `lower fat` cheeses included in the training data, approximately *65.55%* of the dataset. We're going to need to balance the dataset so each of the `FatLevels` above are *50%*.

3.3.2 Data Sparsity

Let's get an understanding of the data sparsity (i.e. *NULL* values).

[7]:
```
display(train_df.describe())
```

```
          CheeseId  MoisturePercent     Organic
count   833.000000       823.000000  833.000000
mean   1557.866747        46.955043    0.094838
std     451.129129         9.557279    0.293167
min     228.000000        12.000000    0.000000
25%    1288.000000        40.000000    0.000000
50%    1535.000000        46.000000    0.000000
75%    1902.000000        52.000000    0.000000
max    2390.000000        88.000000    1.000000
```

[8]:
```
display(train_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 833 entries, 110 to 727
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CheeseId              833 non-null    int64
 1   ManufacturerProvCode  833 non-null    object
 2   ManufacturingTypeEn   833 non-null    object
 3   MoisturePercent       823 non-null    float64
 4   FlavourEn             649 non-null    object
 5   CharacteristicsEn     514 non-null    object
 6   Organic               833 non-null    int64
```

```
 7   CategoryTypeEn         814 non-null    object
 8   MilkTypeEn             832 non-null    object
 9   MilkTreatmentTypeEn    779 non-null    object
 10  RindTypeEn             583 non-null    object
 11  CheeseName             833 non-null    object
 12  FatLevel               833 non-null    object
dtypes: float64(1), int64(2), object(10)
memory usage: 91.1+ KB
```

None

It seems that the `FlavourEn`, `CharacteristicsEn`, `RindTypeEn` have a high amount of *NULL* values. Let's visualize the distribution in the dataset.

3.3.3 Observe Data Sparsity

[9]:
```
heatmap_df =  X_train.isna().reset_index()
heatmap_df.rename(columns = {'index' : 'Index'}, inplace = True)
heatmap_df = heatmap_df.melt(
    id_vars = 'Index',
    value_vars = [col for col in heatmap_df.columns.to_list() if col !=␣
 ↪'Index'],
    var_name = 'Columns',
    value_name = 'IsNull'
)
```

[10]:
```
sparsity_plot = alt.Chart(
        heatmap_df,
        title = alt.TitleParams(f'Figure {fig_number} : Cheese Dataset - Data␣
 ↪Availability', fontSize = 27.5)
).mark_rect().encode(
    x = alt.X(
        'Index:Q', title = '', axis = None
    ),
    y = alt.Y('Columns:N', title = 'Features'),
    tooltip = [alt.Tooltip('Index:Q', title = 'Index')],
    color = alt.Color(
        'IsNull:O',
        scale = alt.Scale(
            domain = [False, True],
            range = ['#000000', '#FFFFFF']
        ),
        legend = alt.Legend(title = 'Null Data')
    ),
).properties(
    height = 500, width = 600,
).configure_axis(labelFontSize = 15, titleFontSize = 20)
```

```
display(sparsity_plot)

fig_number += 1
```

```
alt.Chart(…)
```

3.3.4 Drop Columns From Dataset

From the visualization above, lets exclude `CharacteristicsEn`, `FlavourEn` due to the high level of data sparsity. Let's remove `RindTypeEn` as well. From displaying `train_df.info()` it seems as if the data sparsity is higher than it appears from Figure 2.

We should also drop the `CheeseIds` from our training dataset since these won't be useful for statistical modeling.

[11]:
```
X_train.drop(
    columns = ['CheeseId', 'CharacteristicsEn', 'FlavourEn', 'RindTypeEn'],
    inplace = True
)
```

3.3.5 Observe Feature Types

Let's distinguish the feature types in our dataset.

[12]:
```
describe_df = X_train.describe(include = 'all').T
display(describe_df)
```

```
                      count unique             top freq       mean       std  \
ManufacturerProvCode    833     10              QC  637        NaN       NaN
ManufacturingTypeEn     833      3      Industrial  371        NaN       NaN
MoisturePercent       823.0    NaN             NaN  NaN  46.955043  9.557279
Organic               833.0    NaN             NaN  NaN   0.094838  0.293167
CategoryTypeEn          814      6     Firm Cheese  276        NaN       NaN
MilkTypeEn              832      8             Cow  591        NaN       NaN
MilkTreatmentTypeEn     779      3     Pasteurized  640        NaN       NaN
CheeseName              833    830  Ménestrel (Le)    2        NaN       NaN

                       min   25%   50%   75%   max
ManufacturerProvCode   NaN   NaN   NaN   NaN   NaN
ManufacturingTypeEn    NaN   NaN   NaN   NaN   NaN
MoisturePercent       12.0  40.0  46.0  52.0  88.0
Organic                0.0   0.0   0.0   0.0   1.0
CategoryTypeEn         NaN   NaN   NaN   NaN   NaN
MilkTypeEn             NaN   NaN   NaN   NaN   NaN
MilkTreatmentTypeEn    NaN   NaN   NaN   NaN   NaN
CheeseName             NaN   NaN   NaN   NaN   NaN
```

[13]:
```
objects_df = X_train.describe(include = 'object').T
display(objects_df)
```

```
                      count unique             top freq
ManufacturerProvCode    833     10              QC  637
ManufacturingTypeEn     833      3      Industrial  371
CategoryTypeEn          814      6     Firm Cheese  276
MilkTypeEn              832      8             Cow  591
MilkTreatmentTypeEn     779      3     Pasteurized  640
CheeseName              833    830  Ménestrel (Le)    2
```

[14]:
```
numeric_df = X_train.describe(include = ['int64', 'float64']).T
display(numeric_df)
```

```
                 count       mean       std   min   25%   50%   75%   max
MoisturePercent  823.0  46.955043  9.557279  12.0  40.0  46.0  52.0  88.0
Organic          833.0   0.094838  0.293167   0.0   0.0   0.0   0.0   1.0
```

Numeric Feature The `MoisturePercent` column is a numeric feature.

[15]:
```
numeric_feats = [feat for feat in numeric_df.index if len(X_train[feat].
 ↪unique()) != 2]
display(numeric_feats)
```

```
['MoisturePercent']
```

[16]:
```
fig_number = describe_features(
    effective_df = X_train,
    features = numeric_feats,
    fig_number = fig_number
)
```

```
The distinct values in the MoisturePercent column are :
[52.0, 40.0, 48.0, 55.0, 60.0, 39.0, 50.0, 56.0, 57.0, 46.0, 42.0, 58.0, 37.0,
44.0, 59.0, 88.0, 41.0, 43.0, 33.0, 35.0, 38.0, 27.0, nan, 36.0, 45.0, 61.0,
31.0, 80.0, 68.0, 62.0, 51.0, 64.0, 76.0, 34.0, 74.0, 47.0, 49.0, 29.0, 54.0,
40.3, 32.0, 22.0, 70.0, 86.0, 65.0, 75.0, 26.0, 78.0, 20.0, 23.0, 72.0, 63.0,
49.4, 12.0, 25.0, 53.0, 30.0, 17.0, 47.9, 42.8, 42.6, 83.0, 69.0, 51.7, 21.0]
```

```
alt.Chart(…)
```

Binary Feature The `Organic` column is a binary feature.

[17]:
```
binary_feats = [feat for feat in describe_df.index if len(X_train[feat].
 ↪unique()) == 2]
display(binary_feats)
```

```
['Organic']
```

[18]:
```
fig_number = describe_features(
    effective_df = X_train,
    features = binary_feats,
```

```
    fig_number = fig_number
)
```

```
The distinct values in the Organic column are :
[0, 1]
```

```
alt.Chart(…)
```

Categorical Features The `ManufacturerProvCode`, `ManufacturingTypeEn`, `CategoryTypeEn`, `MilkTypeEn`, `MilkTreatmentTypeEn` columns are categorical features.

[19]:
```
categorical_feats = objects_df[
    (objects_df['unique'] < 0.1 * objects_df['count']) &
    (objects_df['freq'] != 2)
].index.to_list()

display(categorical_feats)
```

```
['ManufacturerProvCode',
 'ManufacturingTypeEn',
 'CategoryTypeEn',
 'MilkTypeEn',
 'MilkTreatmentTypeEn']
```

[20]:
```
fig_number = describe_features(
    effective_df = X_train, features = categorical_feats,
    fig_number = fig_number, sort_by = 'x'
)
```

```
The distinct values in the ManufacturerProvCode column are :
['ON', 'QC', 'BC', 'AB', 'NB', 'NS', 'PE', 'MB', 'NL', 'SK']
```

```
alt.Chart(…)
```

```
The distinct values in the ManufacturingTypeEn column are :
['Industrial', 'Artisan', 'Farmstead']
```

```
alt.Chart(…)
```

```
The distinct values in the CategoryTypeEn column are :
['Semi-soft Cheese', 'Firm Cheese', 'Soft Cheese', 'Fresh Cheese', nan, 'Hard
Cheese', 'Veined Cheeses']
```

```
alt.Chart(…)
```

```
The distinct values in the MilkTypeEn column are :
['Cow', 'Goat', 'Ewe', 'Ewe and Cow', 'Cow and Goat', 'Cow, Goat and Ewe', 'Ewe
and Goat', 'Buffalo Cow', nan]
```

```
alt.Chart(…)
```

```
The distinct values in the MilkTreatmentTypeEn column are :
['Pasteurized', 'Raw Milk', 'Thermised', nan]
```

```
alt.Chart(…)
```

Free-Text Feature The CheeseName column is a free text feature in our dataset. This is going to be fun to tackle. Let's start by printing the values to see what we have to deal with.

[21]:
```
text_feats = objects_df[
        (objects_df['unique'] > 0.1 * objects_df['count']) &
        (objects_df['freq'] == 2)
    ].index.to_list()
display(text_feats)
```

```
['CheeseName']
```

[22]:
```
for feat in text_feats :
    print(f'''The distinct values in the {feat} column are :␣
 ↪\n{list(X_train[feat].unique())}\n''')
```

```
The distinct values in the CheeseName column are :
['Vaquinha  (Portuguese)', 'Gorgonzola (Castello)', 'Tête à Papineau', 'Petit
Rubis (Le)', 'Petites Soeurs (Les)', 'Cheddar 2 ans (Fromagerie Perron)', 'Brie
Normandie double crème', 'Médard (Le)', 'Champfleury (Vaudreuil)', "St.John's
Cow (Portuguese)", "Extra chèvre (L')", 'Chevrier', 'Tablée (La)', 'Sieur Riou-x
(Le)', 'Médaillon et le Tournevent (Le)', 'Brie Vaudreuil Double Crème',
'Cheddar (Mornington Dairy)', 'Trésor du Fumoir', 'Caprice des Saisons', "P'tit
féta", 'Damablanc', "Rayon d'or", "Cheddar L'Autre Versant", 'Menoum!', 'Eweda
cru (Best Baa Dairy)', 'Cheddar (Biobio) - 1 year, 2 years and 3 years',
'Cracked Pepper Verdelait', 'Bocconcini (International Cheese)', 'Savoury Moon',
'Petit Poitou (Le)', 'Cheddar Coaticook', 'Légère Brise du Matin', 'Mozzarella
(Lino)', 'FRESK-O', 'Evanturel', 'Brie double crème Provençal', 'Santa Lucia
Tuma', 'Baluchon', "Alpinois (L')", 'Tomme Ferlend', 'Raclette (Little
Qualicum)', 'Brie double crème Anco', 'Honfleur', 'Mountain Grana', 'Poivroux
(Le)', 'Caciotta (Salerno)', 'Brick St-Guillaume', 'Moine (Le)', 'Cheddar
Jocoeur', 'Wasabi Verdelait', 'Parmesan (Fromagerie Saint-Laurent)', 'Suisse
léger (Fromagerie St-Fidèle)', 'Clandestin (Le)', 'Allégro 17%', "Archange
(L')", 'Capra (Le)', 'Brie Le Petit Champlain', 'Holmestead Feta', 'Santa Lucia
Goat Cheese', 'Avonlea Clothbound Cheddar', 'Chevalier de Lorimier', 'Mistouk
(Le)', 'Cheddar frais Côte-de-Beaupré', 'Fetaccompli', 'Saumuré Coaticook',
'Beddis Blue', 'Doyen (Le)', 'Brie Vaudreuil', 'Tomme des cantons', 'Garlic
Cheddar', 'Cheddar frais (Fromagerie du Matin)', 'Dama-12', 'Cheddar extra-fort
(Fromagerie des Basques)', 'Avalanche', "Grelotins à l'huile aromatisée", 'Brise
du Matin', "Cheddar fort  (l'Ancêtre)", 'Monterey Jack St-Guillaume', 'Bleubry',
'Fresh Chèvre', 'Cheddar (Fromagerie Port-Joli)', 'Monterey Jack (Bothwell)',
'Rondoux pur chèvre', 'Micherolle', 'Rougette de Brigham', 'Asiago (Tre
Stelle)', 'Cheddar (Fromagerie Couland)', 'Petits caprice', "Oka l'artisan",
'Camembert Vaudreuil', 'Seigneur de Tilly (Le)', 'Fridolines', 'Camembert Le
```

Grand Cru', 'Aged Farmhouse', 'San Pareil', 'Casey (Blue Cheese)', 'Monterey Jack (Black Diamond)', "Ange Fourchu (L')", 'Fromage en grains la Chèvrerie Barrousse', 'Ricotta Prestigio', 'Cendrillon (Le)', 'Comox Camembert', "Rivière Rouge d'Oka", 'Cheddar mi-fort (Fromagerie Lemaire)', 'Saint-Honoré', 'Queijo do Pico', 'Mamirolle (Le)', 'Roy Léo', 'Victor et Berthold', 'Grain caprin', 'Cheddar 1 an (Fromagerie Perron)', 'Tarapatapom', 'Gouda', 'Jac le Chevrier', 'Goat Lindsay Bandaged Cheddar (Celebrity International)', 'Féta de chèvre (Anco)', 'Vacherin de Châteauguay', 'Cheddar fort (Fromagerie Saint-Laurent)', 'Mild Goat Cheese', 'Camembert double crème (Damafro)', 'Gré des champs (Le)', 'Kraft LiveActive', 'Camembert Connaisseur', 'Red Hot Chili Pepper Jack', 'Fée des bois', 'Grondines (Les)', 'Scamorza (Salerno)', 'Pacific Rock', 'Barthélemy (Le)', 'Délice', "Cheddar nature (le P'tit Train du Nord)", 'Valida Fort (Le)', 'Grand Cahill (Le)', 'Caronzola', 'Blue Cheese (Rosenborg)', 'Cheddar', 'Cheddar fort (Fromagerie Lemaire)', 'Maasdammer', 'Bocconcini (Scardillo)', 'Raclette Anco', 'Saint-Pierre de Saurel léger', 'Empereur léger', 'Chaliberg léger', 'Cheddars (Black Diamond)', 'Symandre', 'Edam (Scardillo)', 'Rose', 'Ranchero Fresh Cheese', 'Belle du Jersey (La)', "P'tit Saint-Damase léger", 'Laliberté (triple crème)', 'Cheddar fort St-Guillaume', 'Sheep Feta', 'Udderly Organic Goat Feta (Jerseyland Organic)', 'Bocconcini (Tre Stelle)', "Mozzarella 20% (l'Ancêtre)", 'Freddo (Le)', 'Cheddar frais Le Fromage au Village', "L'Héritage", 'Goat Brie (Woolwich)', 'Crottin de chèvre Capriati', 'Saint-Pierre de Saurel', 'Jeune-Coeur', 'Chèvre des Alpes', 'Féta du Domaine', 'Edam (Tres Stelle)', 'Cabriolet (Le)', 'Chevrita', 'Belle Crème', 'Fromage frais assaisonné', 'Chèvre-Noît Bleu de chèvre', 'Mozzarellissima', 'Cru du Clocher (Réserve 2 ans) (Le)', 'St-Charles', 'Du Charme', "Cheddar moyen (l'Ancêtre)", 'Chèvre de Gaspé', 'Saint-Paulin (Damafro)', 'Ash-Ripened Camembert', 'Grana (Jerseyland Organic)', 'Chèvrai (Le)', 'Tomme de Monsieur Séguin (La)', 'Provolone Sette Fette (Tre-Stelle)', 'Cantolait', 'Garlic and Chive Verdelait', 'Camembert de Portneuf', 'Pont Tournant (Le)', 'Raclette Fritz Poivre', 'Burrata', 'Brick (Fromagerie Saint-Laurent)', 'Parmesan Rivièra', 'Bocconcini (Saputo)', 'Cheddar frais (Fromagerie St-Fidèle)', 'Udderly Organic Goat Gouda (Jerseyland Organic)', 'Monnoir (Le)', 'Jensen Cheese (Wilton Cheese Factory)', 'Winterlude Spice Verderlait', 'Chénéville', "Agate de St-Damien (L')", 'Cogruet', 'Cheddar aux fines herbes (Fromagerie Saint-Laurent)', 'Fromage de chèvre Tour de France', 'Soupçon de Bleu (Le)', 'Skouik!', 'Sieur Corbeau des Laurentides', 'Gouda (Sylvan Star Ltd.)', 'Sieur de Duplessis (Le)', 'Brick (Village Cheese)', 'Desneiges (Le)', 'Buchevrette', 'Labneh', 'Barbu (Le)', 'Little Qualicum Feta', 'Péningouin (Le)', 'Corvo Semi-Soft Cheese (Portuguese)', 'Camembert Allégro', 'Mont-Jacob (Le)', 'Raclette Fritz', 'Vlahos Feta', 'Tour St-François (La)', 'Cheddar très fort (Fromagerie Lemaire)', 'Saint-Damase', 'Clef des Champs (La)', "Brie L'Extra", 'Moujadalé', 'Cheddar (Farmers Dairy)', 'Tomme de Gaston', 'Raclette Kingsey', 'Fêtard (Le)', 'Parmesan (Tre Stelle)', 'Sonatine', 'Tintamarre (Le)', 'Cheddar Coaticook non salé', "Bergeronds dans l'huile (Les)", 'Gouda de chèvre (Damafro)', 'Monterey Jack (Village Cheese)', 'Petit Normand (Le)', 'Swiss Alpine Meadow (Village Cheese)', 'Grand Manitou (Le)', 'Chant du Coq (Le)', 'Cheddar (Thornloe)', 'Oka léger', 'Cheddar (Jerseyland Organic)', 'Cheddar sans sel (Fromagerie Boivin)', 'Côte-Sud (Le)', 'St-Antoine (Le)', 'Samuel et Jérémie', 'Cheddar La Chaudière',

'Cheddar (Biobio) - Mild, medium and strong', 'Lavallois (Le)', 'Cabrie', 'Chèvratout', 'Hip Hop', 'Cheddar La Galipette', 'Rathtrevor', 'Cheddar (Bright Brand)', 'Prés de Kildare', 'Fromage Frais', 'Emmental St-Guillaume', 'Cumin Verdelait', 'Forban (Le)', 'Feta Floralpe', 'Fredondaine', 'Halloom', 'Brie Chevalier triple crème', 'Ricotta (Skotidakis)', 'Pleine Lune (Le)', 'Tuma Cheese (Fresh Cheese)(Salerno)', 'Ricotta de Lactosérum Bari', 'Délice des Appalaches', 'Bleu de Brebis de Charlevoix (Le)', 'Démon (Le)', 'Nabulsi (Fromagerie Marie Kadé)', 'Caprichef', 'Bouton de Culotte (Le)', 'Goat Mozarella (Woolwich)', 'Allegretto (Le)', 'Chevrochon', "Gouda (Cheeselady's)", 'Curé-Hébert (Le)', 'Gouda (Ivanhoe)', 'Gouda (Bothwell)', 'Quark cheese (Damafro)', 'Peter', 'Récompense (La)', 'Gouda Anco', 'Tressé (Fromagerie Polyethnique)', 'Haloumi (Fromagerie Polyethnique)', "Épave (L')", 'Jersey du Fjord', 'Fleurmier de Charlevoix (Le)', 'Normandises (Les)', 'Feta (Anco)', "Provoletta (Franco's Brand)", 'Benedictus (Le)', "Chute à l'Ours (La)", 'Féta léger (Danesborg)', 'Sainte-Rose', 'Tortillard', 'Roméo (Le)', 'Montbeil (Le)', 'Petit frais aux canneberges', 'Six Pourcent 6% (Le)', 'Swiss (Central Dairies)', 'Saint-Augustin (Le)', 'Douce Folie', 'Mi Chèvre-Mi Vache', 'Brise des Vignerons (La)', 'Fumambule (Le)', 'Petit Pâturin (Le)', 'Tiger Blue', 'Mouton rouge (Best Baa Dairy)', 'Moutier (Le)', "P'tit Bonheur (Le)", 'Comox Brie', 'Mozzarella (Fiorella)', 'Sorcier de Missisquoi (Le)', "Cheddar en grains (SCA L'île-aux-Grues)", "Canotier de l'Isle", 'Capella', 'Asiago (Thornloe)', 'Chute Chaudière (La)', 'Angelus', "P'tit Basque", "Tomme D'Elles (La)", 'Sauvagine (La)', 'Citadelle', 'Tentation de Laurier (La)', 'Trappeur (Le) Brie Double crème', 'Montagnard (Le)', 'Chèvre', 'Migneron de Charlevoix (Le)', 'Myzithra', 'Brie double crème de Portneuf', 'Suisse au lait cru (Fromagerie Perron)', 'Caprano', 'Mozzarella (Prestigio)', 'Farandole', 'Ayoye!', 'St-Tordu', 'Syrean', "p'tit blanchon (le)", 'Petit Frais (Le)', 'Barbizon (Le)', 'Double Joie (Le)', 'Raclette (Fromagerie Champêtre)', 'Cheddar (Empire Cheese)', 'Caciocavallo fumé (Saputo)', 'Grey Owl', 'Suisse Grubec léger', 'Chute à Michel (La)', "D'Iberville (Le)", 'Valida Doux (Le)', 'Comtomme', 'Feta', 'Tomme des Cantons (Fromagerie 1860 Du Village) Saputo', 'Rumeur (La)', 'Péribonka (Le)', 'Tomme du Haut-Richelieu', 'Blanchon (Le)', 'Suisse Grubec', 'Cheddar frais St-Guillaume', "L'Aubert de Gaspé", 'Cheddar St-Georges', "P'tite Chevrette", 'Vent des îles', 'Cheddar doux Riviera', 'Marie-Charlotte', 'Oka avec champignons', "Extra Aged Gouda (Gort's Gouda)", 'Charlton', 'Eweda (Best Baa Dairy)', "Origine de Charevoix (L')", 'Brie Connaisseur (Damafro)', 'Mozzarella Chèvre (Le)', 'Sirocco', 'Petit lardé (Le)', 'Baladi (Fromagerie Marie Kadé)', 'Polichinel (Le)', 'Allégro 7%', 'Saint-Coeur-de-Marie', 'Grand Duc (Le)', 'Gouda de Saint-Antoine', 'Calendos (Le)', "Cheddar extra-fort (l'Ancêtre)", 'Chèvre des Alpes BIO', 'Okanagan Goat Cheese', 'Ricotta Prestigio léger', 'Fleur des Monts', 'Akawie (Fromagerie Polyethnique)', 'Cheddar médium Riviera', 'Cendré des Prés (Le)', 'Provolone (Tre Stelle)', 'Camembert BIO (Damafro)', 'Cheddar moyen (Fromagerie des Basques)', 'Noyan', 'Ballot (Le)', "Cheddar Bio d'Antan", "Acadiac (L')", 'Capri-Corne', 'Coureur des bois (Le)', 'Rang des Îles (Le)', 'Mozzarella', 'Shredded Parmesan with Romano Cheese', "Foin d'odeur (Le)", 'Troubadour (Le)', "Feuille d'automne", 'Havarti Danesborg léger', 'Montefino frais', "P'tit Bronzé", 'Feta de brebis (Alexis de Portneuf)', 'Krinos Sheep Feta (Shepherd Gourmet)', 'Petit Péché (Le)', 'Tilsit', 'Brebis frais (Best Baa

Dairy)', "Presqu'île (Le)", 'Cheddar Littoral', 'Louis-Joseph Papineau', 'Mozzarella (Mozza Fina)', 'Canadiane', 'Cabrita', 'Cheddar mi-fort St-Guillaume', 'Cheddar en grains et en bloc (Fromagerie Champêtre)', 'Biquet (Le)', 'Scamorza (International Cheese)', 'Métayères (Les)', 'Galarneau', 'Brie Notre-Dame', 'Feta (Jerseyland Organic)', "Bouton d'Or (Le)", 'Chèvre doux', 'Clé des champs', 'Lotbinière (Le)', 'Raclette Fritz Fort', 'Cogruet (Le)', 'Cru du Clocher (Le)', 'Brie Marie-Charlotte', 'Cancre', 'Cheddar Chèvre (Le)', 'Suisse léger (Fromagerie Perron)', "Devil's Rock (Creamy Blue Cheese)", 'Caprice des vents', 'Féta (Troupeau Bénit)', 'Rouet', 'Cheddar frais (Qualité Summum)', 'Cheddar frais (Fromagerie Saint-Laurent)', 'Gaulois de Portneuf (Le)', 'Bûchette de Chèvre (La)', "Brie L'Extra double crème", 'Romano (Ivanhoe)', 'Dame de Coeur (La)', "Labneh dans l'huile", 'Brie 4 Temps', "Hercule de Charlevoix (L')", 'Fior di Latte (Natural Pastures Cheese Company)', 'Heidi', 'Voltigeur', 'Cheddar (Balderson)', 'Ramembert (Best Baa Dairy)', 'Brie Mont-Laurier', 'Mini-Brie', 'Santa Lucia Provolone (International Cheese)', 'Calumet (Le)', 'Double Crème DuVillage', 'Cheddar Blackburn (Le)', 'Tomme Blanche', 'Petite Folie (La)', 'Oka Classique', 'Tomme du Draveur (La)', 'Bercail (Le)', 'Cheddar extra-fort Riviera', 'Alpindon "Gift Of The Alpine"', 'Miranda', 'Chèvre des Neiges (brie triple crème)', 'Cheddar Le Bourgadet doux', 'Fontina Canadien', 'Maître Jules', 'Réserve spéciale 115e', 'Gamin (Le)', 'Oka', 'Parmesan Chèvrerie Dion', 'Montasio (Tres Stelle)', 'Cottage de la Beurrerie du Patrimoine', 'Caciocavallo (Saputo)', 'Brie Chevalier double crème', 'Cheddar frais du jour (Fromagerie Perron)', 'Mozzarella (Salerno)', '14 Arpents (Le)', 'Swiss (Thornloe)', 'Bocké (Le)', 'Fetos Feta (Saputo)', 'Wabassee', 'Trois-Pistoles (Le)', 'Ricotta La Moutonnière', 'Graciosa Semi-Soft Goat Cheese (Portuguese)', 'Mario Hébert (4ième génération)', 'Petit Émile (Le)', 'Mackenzie (Le)', 'Cru des Érables (Le)', 'Féta de Chèvre (Le)', 'Goat Cheddar (Woolwich)', 'Zacharie Cloutier', 'Blanche du Fjord (La)', 'Nabulsi (Fromagerie Polyethnique)', 'Clos Saint-Ambroise', 'Patte Blanche', 'Cheddar extra-fort St-Guillaume', 'Baya', 'Raclette des Appalaches', 'Coeur de Brie', 'Mozzarella (Village Cheese)', 'Champayeur (Le)', 'Triple Cream Camembert', 'Grand Chouffe (Le)', 'Saint-Paulin Fritz léger', 'Vacherin Fri-Charco', 'Suisse (Fromagerie St-Fidèle)', "Sir Laurier d'Arthabaska", "Camembert L'Extra", 'Hilairemontais (Le)', 'Camarades (Des)', 'Mascarpone (Silani)', 'Brie Chevalier', 'Gilles Hébert (3ième génération)', 'Romanello (Salerno)', 'Feta Diodati', 'Parmadammer', 'Camembert (Damafro)', 'Camembert Petit Champlain (Le)', 'Bocconcini (Fiorella)', 'Coteau Hills Creamery Balkan Feta Style', 'Mi-Carême', 'Roubine de Noyan', "Ange Gardien (L')", 'Ste-Élisabeth', 'Fleur de Weedon (La)', 'Grand Délice', 'Bleu Fumé', 'Raclette de Joliette (La)', 'Chèvre Le Grand Cru', 'Windigo', 'Tiguidou', 'Brick (Tre Stelle)', 'Saint-Damase BIO', 'Chèbrie', "Tomme d'Or (Moonstruck Organic)", 'Feta Tradition', 'Saint-Paulin DuVillage', 'Bûchette (La)', 'Princeville', 'Montebello (Le)', 'Suisse macéré au porto (Fromagerie St-Fidèle)', 'Brie Tour de France (Damafro)', 'Friulano (International Cheese)', "Étoile Bleue de Saint-Rémi (L')", 'Petit Bayou (Le)', "Alpin (L')", 'Ste-Anne', 'Gouda (Jerseyland Organic)', 'Golden Blyth', 'Brie Le Grand Cru', "Cheddar vieilli de l'île-aux-Grues", "Cheddar doux (l'Ancêtre)", 'Petit Brie DuVillage', 'Fleur Saint-Michel', "Val-D'Espoir", 'Grand Cheddar Réserve Spéciale', 'Laracam', 'Bleu de la Moutonnière (Le)', "Élite (L')", 'Oo

La La Hot Jill', 'ChamPaître (Le)', 'St-Émile', 'Ermite', 'Doux du Fort (Le)', 'Kunik', 'Ma Manière (Le)', 'Island Bries', 'Ciel de Charlevoix (Le)', 'Secret de Maurice (Le)', 'Frère Jacques', 'Tomme Le Champ Doré', 'Mouton noir', 'Tomme du Manoir affinée au cidre de pomme', 'Rose Haus', 'Cheddar médium (Fromagerie Victoria)', 'Petit Prince (Le)', 'Grand Camembert Vaudreuil', 'White Moon', "Providence d'Oka (La)", 'Rightrou!', 'Fleur de Neige', 'Kobrossi', 'Agnelle de Bayolle', 'Boucané (Le)', 'Goat Mozzarella (Celebrity Interantional)', 'Diable aux Vaches (Le)', 'Trappeur (Le) Camembert Double crème', 'Chèvre frais fermier', 'Saint-Médard', 'Baladi (Fromagerie Polyethnique)', 'Fin Renard (Le)', "Cheddar fumé au bois d'érable", 'Okanagan Double Cream Camembert', 'Suisse (Fromagerie Perron)', 'Cheddar Charlevoix', 'Chanoine', 'Romano (Thornloe)', "Raclette d'Oka", 'Valida Moyen (Le)', 'Cru du Canton (Le)', 'Sentinelle', 'Sieur Colomban', 'Saint-Paulin (Des Basques)', 'Casimir (Le)', 'Fromage en grain', 'Ti-Lou', 'Cabrouet (Le)', 'Brie Petit Connaisseur', 'Mamamia!', 'Fetos Feta léger (Saputo)', 'Féta (Mes Petits Caprices)', 'Naramata Bench Blue', 'Grain de Bayonne', 'Joséphines', 'Annobli', 'Provolone (Silani)', 'Darac', 'Albert Hébert (1ière génération)', 'Tressé (Fromagerie Marie Kadé)', 'Bocconcini (Salerno)', 'Petit Saguenéen (Le)', 'Prés de la Bayonne (Les)', "Ben d'Adon!", 'Douanier (Le)', "St. John's Goat (Portuguese)", 'Cristalia', "Lady Laurier d'Arthabaska", 'Camembert Provençal', 'Knoydart', 'Deo Gratias (Le)', 'Cheddar très fort (Fromagerie St-Fidèle)', 'Provolone (Saputo)', 'Cheddar (Tres Stelle)', 'Sabot de Blanchette (Le)', 'Grand 2 (Le)', 'Cheddar frais (Fromagerie des Basques)', 'Swiss (Bothwell)', 'Brie en brioche', "Dragon's Breath", 'Traditional Feta', 'Populaire (Le)', 'Labneh (Fromagerie Polyethnique)', 'Tomme du Kamouraska (La)', 'Micha', 'Noble (Le)', 'Monterey Jill', "Écume des Rapides (L')", 'Kefalotri / Saganaki', 'Dame du Lac (La)', 'Campagnard', 'Pampille', 'Gigot (Le)', 'Boerenkaas', 'Faisselles (Les)', 'Cheddar (Fromagerie Boivin)', "Parmesan (l'Ancêtre)", 'St-Félicien Lac St-Jean', 'Graviera', 'Cheddar au Porto (Fromagerie Perron)', "Emmental (l'Ancêtre)", 'Farm Cheese', 'Camembert Anco', 'Montefino assaisonné', 'Cacciocavallo (International Cheese)', 'Mini Bocconcini (International Cheese)', 'Bleu Bénédictin', 'Akawie léger (Fromagerie Polyethnique)', 'Déli Chèvre (Le)', "Fontina de l'Abbaye", 'Tre Fratello', 'Fontina Fumé', "Choupet's (Le)", 'Mont Saint-Benoît', 'Cheddar mi-fort (Fromagerie St-Fidèle)', 'Mascarpone (Tre Stelle)', 'Havarti (Dofino)', 'Boules de neige (Les)', 'Havarti Danesborg', 'Goat Feta (Celebrity International)', 'Rassembleu (Le)', 'Swiss', 'La Scala', 'Piekouagami (Le)', 'Pompette (Le)', 'Parmesan (Ivanhoe)', "Gouda (That Dutchman's Farm)", 'Fruitier de Montérégie', 'Akawi', 'Rébellion 1837', 'Bleu Extra-Fort', 'Biquet à la crème (Le)', 'Amourettes - Tomates séchées et basilic', 'Mozzarella (Farmers Cooporative Dairies)', 'Brie double crème (Damafro)', 'Capricook', 'Soeur Angèle (La)', 'Fleur des Monts (Les)', 'Empereur', "Paillasson de l'Isle d'Orléans (Le)", 'Trecce (Saputo)', 'Sao Miguel', 'Zéphyr (Le)', 'Suisse (Fromagerie Saint-Laurent)', 'Suisse (Biobio)', 'Suisse St-Guillaume', 'Voyageur (Le)', 'Capriati', 'Santa Lucia Goat Cheese Brie', 'Donna Hébert (2ième génération)', 'White Grace', 'Temiskaming', '1608 (Le)', 'Asiago (Ivanhoe Cheese)', 'Saint-Raymond (Le)', 'Brebiouais', "P'tit Diable", 'Courtenay Cheddar', 'Tomme de chèvre (Damafro)', 'Niagara Gold', 'Neige de Brebis (Le)', "Évanjules (L')", 'Brin de gouda', 'Tomme de Grosse-Île', "Feta (Gort's Gouda)", 'Tortillons (Les)

```
(Fromagerie Boivin)', "Ricotta (l'Ancêtre)", 'Porto Bleu (Le)', 'Paillot de
chèvre', 'Napoléon (Le)', 'Chevrai (Woolwich)', 'Raclette affinée au cidre de
pommes (Les Dépendances du Manoir)', 'Ricotta (Silani)', "Élan 7% (L')", 'Boules
de Chèvre', 'Cantonnier', 'Electric Blue', 'Feta (Black Diamond)', 'Parmesan
(Village Cheese)', 'Ricotta de chèvre', 'Tomme du Maréchal', 'Chèvre fin', 'Goat
Cups (Celebrity International)', 'Borgonzola (Tre Stelle)', 'Cheddar assaisonné
St-Guillaume', 'Chevrino', "Chèvre D'Art", 'Kénogami (Le)', 'Rondoux triple
crème', "Petit frais aux noix et sirop d'érable", 'Réserve La Pérade',
'Amourettes - Fines herbes', 'Cheddar (Bothwell)', 'Quark (Best Baa Dairy)',
'Mascarpone (Damafro)', 'Gavroche', 'Ricotta Saint-Benoît', 'Brie Mme Clément',
'Ste-Geneviève', "Grand Camembert l'Extra", 'Couventine (Le)', 'Monterey Jack
(Fromagerie Boivin)', 'Rosé du Saguenay', 'Raclette Griffon', 'Corsaire',
'Suisse macéré au Cidre de pommes', 'Oh Chiiz', 'Cheese Curds', 'Feta
(Danesborg)', 'Serra  (Portuguese Cheese Company)', 'Shinglish', 'Brebiane',
'Vacherin Fritz Kaiser', 'Saint-Félix', 'Fou Raide (Le)', 'Mon précieux', 'Geai
Bleu (Le)', 'Roche Noire (La)', 'Perle du Littoral (La)', 'Duo du Paradis',
'Cheddar léger 6% (Fromagerie Gilbert)', 'Tipsy Jill', 'Vieux Charlevoix',
'Chèvre des neiges', 'Belle-Mère (La)', 'Rondoux double crème', 'Féta La
Moutonnière', 'Cheddar doux (Fromagerie Saint-Laurent)', 'Feta (Ivanhoe)',
'Gouda (Damafro)', 'Ménestrel (Le)', 'Pikauba', 'Gouda (Biobio)', 'Memphré
(Le)', 'Cheddar fort (Fromagerie St-Fidèle)', 'Cheddar fort Riviera', 'Fou du
Roy (Le)', 'Parmesan (Biobio)', 'Romano (Tres Stelle)', 'Sheep In The Meadow
(Best Baa Dairy)', 'Cheddar Riviera en grains', 'Pont Blanc (Le)', 'Caprice des
Cantons', 'Bûchette et Pyramide Mes Petits Caprices', 'Kingsberg', 'Guillaume
Tell (Le)', 'Cheddar doux St-Guillaume', 'Madame Chèvre Elite', 'Mozzarella
(Coaticook)', 'Brie (Damafro)', 'Biobio fromage 7%', 'Météorite', 'Tomme de
brebis (Fromagerie Couland)', 'Rivière blanche', 'Cheddar Beauceron', 'Frugal',
'Swiss (Farmers Cooperative Dairy)', 'Marquis de Témiscouata', 'Cheddar doux
(Fromagerie St-Fidèle)', 'Alfred Le Fermier']
```

3.4 Preprocessing The Data

3.4.1 Transformation Pipelines

We're going to transform the feature types with respective transformation pipelines.

3.4.2 Binary Transformer

To begin, we perform imputation on the binary feature with the `most_frequent` value to replace missing values. Then we use `OneHotEncoder()` to numerically encode each binary value (i.e. *True*, *False*). Note that we only need to keep a single binary column and can drop the other.

[23]:
```
binary_transformer = make_pipeline(
    SimpleImputer(strategy = 'most_frequent'),
    OneHotEncoder(dtype = int, handle_unknown = 'error', drop = 'if_binary')
)
```

3.4.3 Numeric Transformer

We perform imputation on numeric features with the `median` value to replace missing values. Then we will standardize the numeric features to set sample mean to *0* and standard deviation to *1*.

[24]:
```
numeric_transformer = make_pipeline(
    SimpleImputer(strategy = 'median'),
    StandardScaler()
)
```

3.4.4 Categorical Transformer

We perform imputation on categorical features with the `most_frequent` value to replace missing values. Then we use `OneHotEncoder()` to numerically encode each categorical value.

[25]:
```
categorical_transformer = make_pipeline(
    SimpleImputer(strategy = 'most_frequent', fill_value = 'missing'),
    OneHotEncoder(dtype = int, handle_unknown = 'ignore')
)
```

3.4.5 Free-Text Transformer

We apply the `CountVectorizer()` tool on the free-text feature with the `most_frequent` value and convert the text messages to a matrix of word counts. Each text message is assigned a row and each column represents a word in the dataset vocabulary. The values in the matrix represents the frequency of occurance of the word.

[26]:
```
text_transformer = make_pipeline(
    CountVectorizer(binary = True)
)
```

3.4.6 Column Transformer

We map the feature types to the transformer pipelines made above and drop the remainder of the dataset columns.

[27]:
```
preprocessor = make_column_transformer(
    # Preprocessing Pipelines
    (binary_transformer, binary_feats),
    (numeric_transformer, numeric_feats),
    (categorical_transformer, categorical_feats),
    (text_transformer, text_feats[0]),
    remainder = 'drop'
)
```

4 ML Models

We will be implementing our preprocessor pipelines with multiple classifiers and viewing the results of each model.

We will be training our dataset with the following Classification ML models.

- *Logistic Regression* Classification
- *Decision Tree* Classifier
- *Random Forest* Classifier
- *K Nearest Neighbors (k-NN)* Classifier
- *Support Vector Machines (SVM)* Classifier

This set of classifiers includes

- Interpretable Modelling (i.e. *Logistic Regression*)
- Rule-Based Algorithms with If-Else Statements (i.e. *Decision Tree*, *Random Forest*)
- Similarity Based Models (i.e. *k-NN*, *SVM*)

[28]:
```
models = {
    'Logistic Regression' : {
        'pipeline' : make_pipeline(
            preprocessor, LogisticRegression(random_state = 77, class_weight =␣
↪'balanced')
        )
    },
    'Decision Tree' : {
            'pipeline' : make_pipeline(
                preprocessor, DecisionTreeClassifier(random_state = 77,␣
↪class_weight = 'balanced')
            )
        },
    'Random Forest' :
        {
            'pipeline' : make_pipeline(
                preprocessor, RandomForestClassifier(random_state = 77,␣
↪class_weight = 'balanced')
            )
        },
    'kNN' :
        {
            'pipeline' : make_pipeline(
                preprocessor, KNeighborsClassifier()
            )
        },
    'RBF SVC' :
        {
            'pipeline' : make_pipeline(
                preprocessor, SVC(random_state = 77, class_weight = 'balanced')
```

```
            )
        }
}
```

4.1 Baseline Model

We will be comparing our classifier models to a *DummyClassifier* estimator using *strategy = 'most_frequent'*.

[29]:
```
dummy_pipe = make_pipeline(
    preprocessor,
    DummyClassifier(random_state = 77, strategy = 'most_frequent')
)
```

4.2 Evaluation Metrics

We will be comparing our model performance based on the following metrics :

- Accuracy
  - "Percentage of Predictions Which Are True"
- Precision
  - "Percentage of Positive Predictions Which Are True"
- F1 Score
  - Combined Score of : "Percentage of Positive Predictions Which Are True" "Percentage of All Positive Examples Which Are Positive Predictions"

4.2.1 Exclusion of Recall

We have excluded Recall from the Evaluation Metrics due to our interest in making the most out of the investment decisions we take as an organization.

In other words, we want to determine if a particular combination of features will yield a successful cheese product. The combinations we miss are unfortunate, but we want to prioritize the reduction (elimination?) of *False Negative* predictions.

[30]:
```
scoring_dict = {
    'accuracy' : make_scorer(accuracy_score),
    'precision' : make_scorer(precision_score, pos_label = 'lower fat'),
    'f1' : make_scorer(f1_score, pos_label = 'lower fat'),
}
```

[31]:
```
dummy_df = pd.DataFrame(
    cross_validate(
        estimator = dummy_pipe, cv = 10,
        X = X_train, y = y_train,
        scoring = scoring_dict,
        return_train_score = True
    )
).mean().apply(
```

```
    lambda x : round(x, 4)
).rename('Dummy Classifier').to_frame().T.reset_index().rename(
    columns = {'index' : 'model'}
)
display(dummy_df)
```

```
              model  fit_time  score_time  test_accuracy  train_accuracy  \
0  Dummy Classifier    0.0282      0.0165         0.6555          0.6555

   test_precision  train_precision  test_f1  train_f1
0          0.6555           0.6555   0.7919    0.7919
```

[32]:
```
scores_df = pd.DataFrame()

for model_name, model in models.items():

    model['scores'] = pd.DataFrame(
        cross_validate(
            estimator = model['pipeline'], cv = 10,
            X = X_train, y = y_train,
            return_train_score = True,
            scoring = scoring_dict
        )
    ).mean().apply(
        lambda x : round(x, 4)
    ).rename(
        model_name
    ).to_frame().T.reset_index().rename(
        columns = {'index' : 'model'}
    )

    scores_df = pd.concat([scores_df, model['scores']], axis = 0)
```

[33]:
```
# We Need to Concatenate the Classifier Scores to the Baseline Model Score.
scores_df = pd.concat([dummy_df, scores_df], axis = 0)
scores_df = scores_df.rename(
    columns = {
        col : col.replace('test', 'validation').replace('time', 'time (s)') ␣
 ↪for col in scores_df.columns.to_list()
    }
)
display(scores_df)
```

```
                 model  fit_time (s)  score_time (s)  validation_accuracy  \
0    Dummy Classifier        0.0282          0.0165               0.6555
0  Logistic Regression       0.0597          0.0142               0.8067
0        Decision Tree       0.0367          0.0141               0.8139
```

```
0        Random Forest        0.3105          0.0279               0.8366
0                  kNN        0.0283          0.0201               0.7934
0              RBF SVC        0.0678          0.0176               0.8294

   train_accuracy  validation_precision  train_precision  validation_f1  \
0          0.6555                0.6555           0.6555         0.7919
0          0.9198                0.8685           0.9615         0.8490
0          1.0000                0.8684           1.0000         0.8552
0          1.0000                0.8360           1.0000         0.8826
0          0.8609                0.8203           0.8679         0.8473
0          0.9160                0.8809           0.9559         0.8677

   train_f1
0    0.7919
0    0.9373
0    1.0000
0    1.0000
0    0.8975
0    0.9344
```

[34]:
```
fig_number = get_scores_chart(
    scores_df = scores_df,
    scoring = 'fit_time',
    fig_number = fig_number
)
```

```
                model    score_type   score
0     Dummy Classifier  fit_time (s)  0.0282
1  Logistic Regression  fit_time (s)  0.0597
2        Decision Tree  fit_time (s)  0.0367
3        Random Forest  fit_time (s)  0.3105
4                  kNN  fit_time (s)  0.0283
```

```
alt.Chart(…)
```

We can see from the figure that the *Random Forest* had a much greater fitting time than the other classifiers, whereas the *k-NN* classifier model had the least fitting time. It's worthwhile to note that all of the classifiers had a relatively similar fitting time, aside from the *Random Forest* model.

[35]:
```
fig_number = get_scores_chart(
    scores_df = scores_df,
    scoring = 'precision',
    fig_number = fig_number
)
```

```
                model       score_type   score
6     Dummy Classifier  train_precision  0.6555
7  Logistic Regression  train_precision  0.9615
8        Decision Tree  train_precision  1.0000
```

```
9          Random Forest  train_precision  1.0000
10                   kNN  train_precision  0.8679
```

`alt.Chart(…)`

We can see in this figure that the best *precision* validation score is from the *RBF SVC* model. We notice that the *Dummy Classifier* model had the lowest *precision* validation score. So far, the *RBF SVC* seems like a promising contender for hyperparameter optimization.

[36]:
```
fig_number = get_scores_chart(
    scores_df = scores_df,
    scoring = 'f1',
    fig_number = fig_number
)
```

```
                  model score_type   score
6       Dummy Classifier   train_f1  0.7919
7    Logistic Regression   train_f1  0.9373
8          Decision Tree   train_f1  1.0000
9          Random Forest   train_f1  1.0000
10                   kNN   train_f1  0.8975
```

`alt.Chart(…)`

We can see in this figure that the highest *F1* validation score is from the *Random Forest* model. The second highest *F1* validation score is from the *RBF SVC* model. The *Dummy Classifier* baseline model once again had the lowest *F1* validation score.

Given the high fitting time of the *Random Forest* model and our emphasis on *precision*, we will be selecting the *RBF SVC* model for hyperparameter optimization.

4.3 RBF SVC Model

4.3.1 Hyperparameter Optimization

Let's optimize the `C` and `gamma` hyperparameters in our model. We will find the best hyperparameters with `GridSearchCV`. We're going to iterate through and exhaust the hyperparameter possibilities since we only have `5^2 = 25` combinations.

We would like to be confident with our cheese product `fat level` predictions to a thorough extent. This will require only slightly more computing resources, which we can run in parallel with `n_jobs = -1`.

[37]:
```
# Handle Case Where Dataset Doesn't Contain Positive Predictions

import warnings
warnings.filterwarnings('ignore')
```

[38]:
```
param_grid = {
    "svc__C" : [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    "svc__gamma" : [0.001, 0.01, 0.1, 1, 10, 100, 1000]
}
```

```
svc_search = GridSearchCV(
    estimator = models['RBF SVC']['pipeline'],
    param_grid = param_grid,
    cv = 10, scoring = scoring_dict['precision'],
    return_train_score = True, n_jobs = -1
)

svc_search.fit(X_train, y_train)
display(svc_search)
```

```
GridSearchCV(cv=10,
             estimator=Pipeline(steps=[('columntransformer',
 ↪ColumnTransformer(transformers=[('pipeline-1',
 ↪Pipeline(steps=[('simpleimputer',
 ↪          SimpleImputer(strategy='most_frequent')),
 ↪         ('onehotencoder',
 ↪          OneHotEncoder(drop='if_binary',
 ↪                        dtype=<class 'int'>))]),
 ↪['Organic']),
 ↪('pipeline-2',
 ↪Pipeline(steps=[('simpleimputer',
 ↪          SimpleImputer(strategy='median'))…
 ↪'MilkTreatmentTypeEn']),
 ↪('pipeline-4',
 ↪Pipeline(steps=[('countvectorizer',
 ↪          CountVectorizer(binary=True))]),
 ↪'CheeseName')])),
                                       ('svc',
                                        SVC(class_weight='balanced',
                                            random_state=77))]),
```

```
            n_jobs=-1,
            param_grid={'svc__C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
                        'svc__gamma': [0.001, 0.01, 0.1, 1, 10, 100, 1000]},
            return_train_score=True,
            scoring=make_scorer(precision_score, pos_label=lower fat))
```

[39]:
```
svc_df = pd.DataFrame(svc_search.cv_results_)

display(svc_df.head())
```

```
   mean_fit_time  std_fit_time  mean_score_time  std_score_time param_svc__C  \
0       0.083696      0.000698         0.017868        0.000607        0.001
1       0.083724      0.000310         0.017794        0.000271        0.001
2       0.083724      0.000607         0.017838        0.000267        0.001
3       0.083231      0.000460         0.017698        0.000167        0.001
4       0.083482      0.000522         0.017761        0.000206        0.001

  param_svc__gamma                                     params  split0_test_score  \
0            0.001  {'svc__C': 0.001, 'svc__gamma': 0.001}                0.0
1             0.01   {'svc__C': 0.001, 'svc__gamma': 0.01}                0.0
2              0.1    {'svc__C': 0.001, 'svc__gamma': 0.1}                0.0
3                1      {'svc__C': 0.001, 'svc__gamma': 1}                0.0
4               10     {'svc__C': 0.001, 'svc__gamma': 10}                0.0

   split1_test_score  split2_test_score  …  split2_train_score  \
0                0.0                0.0  …                 0.0
1                0.0                0.0  …                 0.0
2                0.0                0.0  …                 0.0
3                0.0                0.0  …                 0.0
4                0.0                0.0  …                 0.0

   split3_train_score  split4_train_score  split5_train_score  \
0            0.654667            0.654667            0.654667
1            0.654667            0.654667            0.654667
2            0.654667            0.654667            0.654667
3            0.654667            0.654667            0.654667
4            0.654667            0.654667            0.654667

   split6_train_score  split7_train_score  split8_train_score  \
0               0.656               0.656               0.656
1               0.656               0.656               0.656
2               0.656               0.656               0.656
3               0.656               0.656               0.656
4               0.656               0.656               0.656

   split9_train_score  mean_train_score  std_train_score
0               0.656            0.4588         0.300356
1               0.656            0.4588         0.300356
```

```
2               0.656           0.4588          0.300356
3               0.656           0.4588          0.300356
4               0.656           0.4588          0.300356

[5 rows x 32 columns]
```

Best Model

[40]:
```
best_model = svc_search.best_estimator_

best_svc = svc_search.best_params_
print(f'The best value of C is {best_svc["svc__C"]} and the best value of gamma␣
 ↪is {best_svc["svc__gamma"]}.')

best_svc_score = round(svc_search.best_score_, 2)
print(f'The best validation precision is {best_svc_score}.')
```

```
The best value of C is 0.1 and the best value of gamma is 0.01.
The best validation precision is 0.88.
```

4.3.2 Score Distribution

Let's look at the score distributions for the different hyperparameter combinations.

[41]:
```
svc_plot_df = pd.melt(
    frame = svc_df,
    id_vars = ['param_svc__gamma', 'param_svc__C'],
    var_name = 'score_type', value_name = 'precision',
    value_vars = ['mean_train_score', 'mean_test_score']
)

display(svc_plot_df)
```

```
   param_svc__gamma param_svc__C        score_type  precision
0             0.001        0.001  mean_train_score   0.458800
1              0.01        0.001  mean_train_score   0.458800
2               0.1        0.001  mean_train_score   0.458800
3                 1        0.001  mean_train_score   0.458800
4                10        0.001  mean_train_score   0.458800
..              …           …               …        …
93              0.1         1000   mean_test_score   0.852853
94                1         1000   mean_test_score   0.717279
95               10         1000   mean_test_score   0.657860
96              100         1000   mean_test_score   0.657052
97             1000         1000   mean_test_score   0.657052

[98 rows x 4 columns]
```

[42]:
```
# Create Altair Chart.
svc_plot = alt.Chart(
    svc_plot_df[(svc_plot_df['score_type'] == 'mean_test_score')],
    title = alt.TitleParams(
        text = f'Figure {fig_number} : RBF SVC Model Precision Scores',
        subtitle = ['Hyperparameter Tuning for C and gamma'],
        anchor = 'start', fontSize = 25, subtitleFontSize = 20
    )
).mark_circle().encode(
    x = alt.X('param_svc__gamma:O', title = 'gamma'),
    y = alt.Y('param_svc__C:O', title = 'C'),
    color = alt.Color(
        'precision:Q', title = 'Precision',
        scale = alt.Scale(
            scheme = 'viridis', reverse = True,
            domain = [
                svc_plot_df[
                    svc_plot_df['score_type'] == 'mean_test_score'
                ]['precision'].min(),
                svc_plot_df[
                    svc_plot_df['score_type'] == 'mean_test_score'
                ]['precision'].max()
            ]
        )
    ),
    size = alt.Size(
        'precision:Q', title = 'Precision',
        scale = alt.Scale(
            domain = [
                svc_plot_df[
                    svc_plot_df['score_type'] == 'mean_test_score'
                ]['precision'].min(),
                svc_plot_df[
                    svc_plot_df['score_type'] == 'mean_test_score'
                ]['precision'].max()
            ]
        )
    ),
    tooltip = [alt.Tooltip('precision:Q', title = 'Precision')]
).properties(
    width = 800, height = 500,
).configure_axis(
    labelFontSize = 15, titleFontSize = 17.5
).configure_title(
    fontSize = 25
)
```

```
fig_number += 1

display(svc_plot)
```

```
alt.Chart(…)
```

4.4 Test Data

Let's start by looking at the evaluation metrics for the best *RBF SVC* classifier.

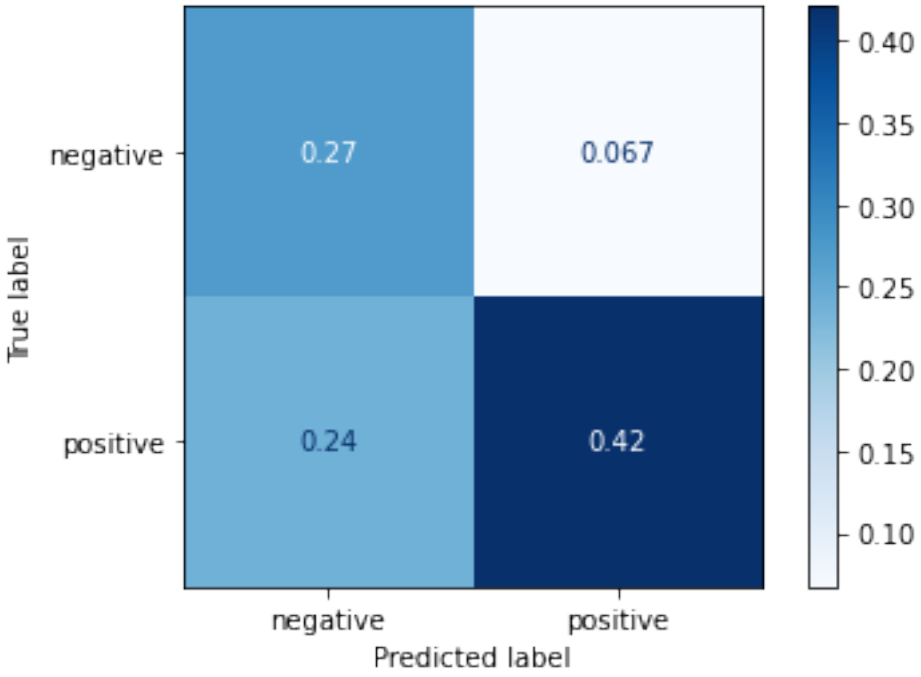
[43]:
```
## We need to drop the columns not used in the Machine Learning pipeline.

X_test.drop(
    columns = ['CheeseId', 'CharacteristicsEn', 'FlavourEn', 'RindTypeEn'],
    inplace = True
)
```

[44]:
```
svm_fatlevels_report = classification_report(
    y_true = y_test,
    y_pred = svc_search.predict(X_test),
    target_names = ['Lower Fat' if x == 'lower fat' else 'Higher Fat' for x in␣
 ↪svc_search.classes_]
)

print(svm_fatlevels_report)
```

```
              precision    recall  f1-score   support

  Higher Fat       0.53      0.80      0.64        71
   Lower Fat       0.86      0.64      0.73       138

    accuracy                           0.69       209
   macro avg       0.70      0.72      0.69       209
weighted avg       0.75      0.69      0.70       209
```

[45]:
```
svm_fatlevels_cm = ConfusionMatrixDisplay(
    confusion_matrix = confusion_matrix(
        y_true = y_test, y_pred = best_model.predict(X_test),
        normalize = 'all'
    ),
    display_labels = ['negative', 'positive']
)

svm_fatlevels_cm.plot(cmap = 'Blues')
```

[45]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fe0669c2a30>
```



We have highly reduced our *False Positive* predictions to *6.7%* with the *RBF SVC* model. This yields a *precision* score of *86%*.

Note that the *False Negatives* are much higher at *24%* of our predictions. This yields a lower *F1* score of *73%*

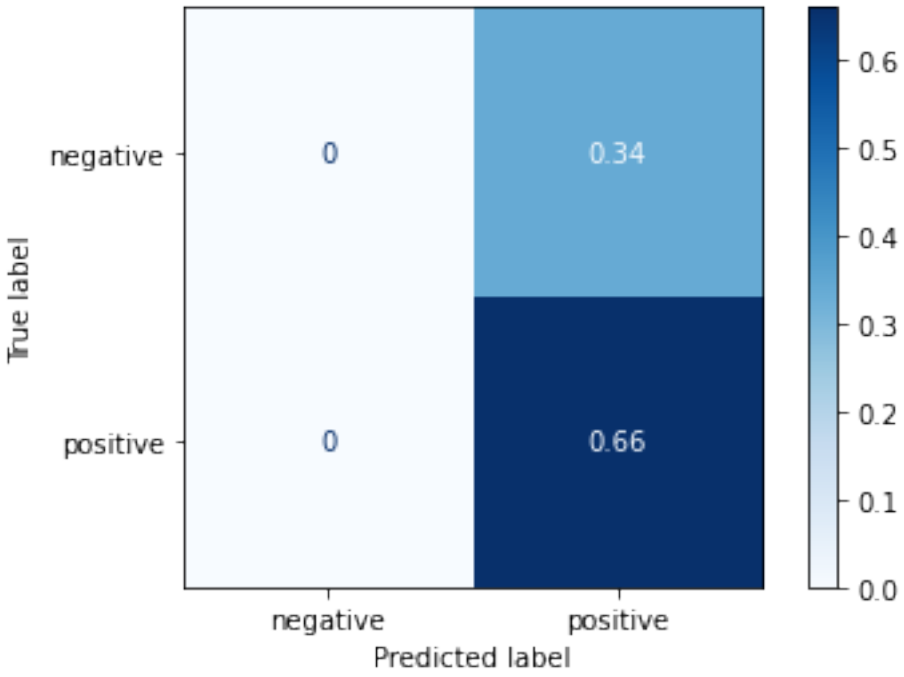
[46]:
```
baseline_fatlevels_report = classification_report(
    y_true = y_test,
    y_pred = dummy_pipe.fit(X_train, y_train).predict(X_test),
)

print(baseline_fatlevels_report)
```

```
              precision    recall  f1-score   support

  higher fat       0.00      0.00      0.00        71
   lower fat       0.66      1.00      0.80       138

    accuracy                           0.66       209
   macro avg       0.33      0.50      0.40       209
weighted avg       0.44      0.66      0.53       209
```

[47]:
```
baseline_fatlevels_cm = ConfusionMatrixDisplay(
    confusion_matrix = confusion_matrix(
```

```
        y_true = y_test, y_pred = dummy_pipe.fit(X_train, y_train).
↪predict(X_test),
        normalize = 'all'
    ),
    display_labels = ['negative', 'positive']
)

baseline_fatlevels_cm.plot(cmap = 'Blues')
```

[47]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fe064abed30>



The *Dummy Classifier* baseline model was unable to make negative predictions. This is rather peculiar and yields a lower *precision* score of *66%*. Interestingly, the perfect *recall* score yields a higher *F1* score of *80%*.

5 Discussion

From this investigation, we were able to observe features which contribute to different fat levels in *Canadian* cheeses and use this in training a Machine Learning classifier to predict whether a cheese will be `lower fat` or `higher fat`.

5.1 Further Improvements

In order to improve the model performance, something which can be explored is the inclusion of the `CharacteristicsEn` and `FlavourEn` columns. These can be transformed with `CountVectorizer()` and we can see how the data sparsity in these columns affects the training and validation scores.

In addition to this, we can look at the `RindTypeEn` column and investigate the feature type and value distributions. There was some peculiar data sparsity here, which was apparent from displaying `train_df.info()`, however filtering the dataframe for *NULL* values didn't register the same data. Exploring this feature may be highly valuable for improving model performance across evaluation metrics.

5.2 Concluding Remarks

We were able to train a *RBF SVC* classifier which performed with *86%* precision. This will aid us in our larger goal of reducing the risk of mass manufacturing unsuccessful cheese products. Since we are predicting a low percentage of *False Negatives* at *6.7%*, we are able to be more confident with the bets that we make. It will be highly beneficial to deploy this model in evaluating test cheese products. If we provide this as a tool to manufacturing experts with extensive domain knowledge, we'll be able to develop a streamlined cheese production practice.

5.3 Citations

These resources provide the theory and code segments for the *ML* exploration in this notebook.

- IBM - What is Machine Learning?
- Introduction to Machine Learning
  - Assignment 5
  - Assignment 6
  - Assignment 7
  - Assignment 8