final_project

March 5, 2024

Title: “Predicting Fat Levels in Canadian Cheeses Through Supervised Learning Analysis”

Introduction:

This original dataset had been taken from the Government of Canada’s Open Government Portal but was unfortunately taken down. The data were obtain from Kaggle and follows an Open Government Licence (Canada). I am using ‘cheese.csv’ for my final Machine Learning project. I am dropping three categorical columns namely, “FlavourEn”, “CharacteristicsEn”, “CheeseName” as they have to be transformed with CountVectorizer which is optional to use. My goal for the project is to use classification to predict the different cheese fat levels based on various features, and answer the fundamental question: “Which features in the dataset contribute to the low or high cheese levels?” I think that the question I am asking here, requires lot of understanding of the dataset and reveals new patterns and main contributing features to the different cheese levels. within the framework of supervised machine learning, I believe that this question will be best answered using classification.

[106]:
```python
# Importing necessary libraries
import altair as alt
import graphviz
import numpy as np
import pandas as pd
import string
from sklearn import tree
from sklearn.dummy import DummyClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, cross_validate
from sklearn.neighbors import KNeighborsClassifier
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.preprocessing import (
    FunctionTransformer,
    Normalizer,
    OneHotEncoder,
    StandardScaler,
    normalize,
    scale)
```

```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score
from sklearn.svm import SVC, SVR

from scipy.stats import lognorm, loguniform, randint
```

[107]:
```
cheese_df = pd.read_csv("data/cheese_data.csv")
# Set the 'CheeseId' column as the index
cheese_df.set_index('CheeseId', inplace=True)
cheese_df.head()
```

[107]:
```
         ManufacturerProvCode ManufacturingTypeEn  MoisturePercent  \
CheeseId
228                        NB           Farmstead             47.0
242                        NB           Farmstead             47.9
301                        ON          Industrial             54.0
303                        NB           Farmstead             47.0
319                        NB           Farmstead             49.4

                                                FlavourEn  \
CheeseId
228                                        Sharp, lactic
242                    Sharp, lactic, lightly caramelized
301                                Mild, tangy, and fruity
303       Sharp with fruity notes and a hint of wild honey
319                                          Softer taste

                                          CharacteristicsEn  Organic  \
CheeseId
228                                                Uncooked        0
242                                                Uncooked        0
301       Pressed and cooked cheese, pasta filata, inter…        0
303                                                     NaN        0
319                                                     NaN        1

            CategoryTypeEn MilkTypeEn MilkTreatmentTypeEn   RindTypeEn  \
CheeseId
228            Firm Cheese        Ewe            Raw Milk  Washed Rind
242       Semi-soft Cheese        Cow            Raw Milk  Washed Rind
301            Firm Cheese        Cow          Pasteurized          NaN
303          Veined Cheeses        Cow            Raw Milk          NaN
319       Semi-soft Cheese        Cow            Raw Milk  Washed Rind

                    CheeseName   FatLevel
CheeseId
228          Sieur de Duplessis (Le)  lower fat
```

```
242                     Tomme Le Champ Doré  lower fat
301       Provolone Sette Fette (Tre-Stelle)  lower fat
303                           Geai Bleu (Le)  lower fat
319                               Gamin (Le)  lower fat
```

[108]:
```
# Dropping unnecessary columns from df
columns_to_drop = ["FlavourEn", "CharacteristicsEn", "CheeseName"]
cheese_df_new =  cheese_df.drop(columns_to_drop,axis=1)
print(cheese_df_new.head())

# Summary statistics of the cheese dataset
cheese_df_new.info()
```

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

            CategoryTypeEn MilkTypeEn MilkTreatmentTypeEn   RindTypeEn  \
CheeseId
228            Firm Cheese        Ewe            Raw Milk  Washed Rind
242        Semi-soft Cheese       Cow            Raw Milk  Washed Rind
301            Firm Cheese        Cow         Pasteurized          NaN
303          Veined Cheeses       Cow            Raw Milk          NaN
319        Semi-soft Cheese       Cow            Raw Milk  Washed Rind

           FatLevel
CheeseId
228       lower fat
242       lower fat
301       lower fat
303       lower fat
319       lower fat
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1042 entries, 228 to 2391
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ManufacturerProvCode  1042 non-null   object
 1   ManufacturingTypeEn   1042 non-null   object
 2   MoisturePercent       1028 non-null   float64
 3   Organic               1042 non-null   int64
 4   CategoryTypeEn        1019 non-null   object
 5   MilkTypeEn            1041 non-null   object
```

```
 6   MilkTreatmentTypeEn   977 non-null    object
 7   RindTypeEn            721 non-null    object
 8   FatLevel              1042 non-null   object
dtypes: float64(1), int64(1), object(7)
memory usage: 81.4+ KB
```

[109]:
```python
cheese_df_new.describe()
```

[109]:

	MoisturePercent	Organic
count	1028.000000	1042.000000
mean	47.069747	0.095010
std	9.592647	0.293369
min	12.000000	0.000000
25%	40.000000	0.000000
50%	46.000000	0.000000
75%	52.000000	0.000000
max	92.000000	1.000000

[110]:
```python
# Checking for null values in the dataset
null_values_df = cheese_df_new.isnull().sum()
print("Null values in the dataframe: \n", null_values_df)
```

```
Null values in the dataframe:
 ManufacturerProvCode      0
ManufacturingTypeEn       0
MoisturePercent          14
Organic                   0
CategoryTypeEn           23
MilkTypeEn                1
MilkTreatmentTypeEn      65
RindTypeEn              321
FatLevel                  0
dtype: int64
```

Exploratory Data Analysis:

About dataset: The cheese dataset has 13 columns and 1042 rows in total highlighting diffrent features like Cheese province origin, cheese moisture percent etc. to predict a target column having different fat levels of cheese. The dataset has more categorical features than numerical features.

Question 1. Discuss any challenges or peculiarities of the dataset. Challenges in this dataset are not many, but on seeing in depth highlights various patterns between features and the target variable.

Question 2. Are there any inconsistencies or errors in the data that need to be addressed? The summary statistics of this dataset highlights various NaN missing values that need to be eliminated.

Question 3. How much data is missing in the dataset? Are there any patterns in the missing data? The summary statistics of this dataset highlights various NaN missing values that need to be eliminated like MoisturePercent has 14 missing values, RindTypeEn has 321 missing values, and MilkTreatmentTypeEn has 65 mising values.

Question 4. Are there any obvious relationships between features or between the target and any of the features? Yes there are obvious relationships between target FatLevel and the features like MoisturePercent etc.

Question 5. What is the distribution of the data? Are the datapoints skewed in some way? The density plot of MoisturePercent numerical column shows that the data hsas normal distribution, and no datapoints are skewed.

Question 6. Do you need to deal with class imbalance? Explain your answer in your own words. Yes, there exists a class imabalance in the target column FatLevel, where the lower fat cheese are higher in value than high fat cheese, and its a case of binary classification.

Question 7. Specify the metrics that will be used to evaluate the success of your project (accuracy, precision, recall, F1-score, ROC-AUC, etc.). Discuss why these metrics are chosen and how they relate to the project's goal. Since this is a binary classification, I think that accuracy of the model, F1 Score will be valuable metrics for this case. As high accuracy and high f1 score suggest that that model is a better choice for these ML problems.

[111]:
```python
# Creating EDA to understand the data
# Visualizing "c" vs target variable column "FatLevel" using Altair
Moisture_vs_Fat_plot =  alt.Chart(cheese_df_new).mark_line().encode(
    x=alt.X('MoisturePercent:Q', bin=alt.Bin(maxbins=30), title='Moisture␣
 ↪Percent'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Precentage of Moisture vs Fat Level of the Cheese',
    width=500
)



Moisture_vs_Fat_plot
```

[111]: alt.Chart(…)

[112]:
```python
# Creating EDA to understand the data
# Visualizing "CategoryTypeEn" vs target variable column "FatLevel" using Altair
Category_vs_Fat_plot =  alt.Chart(cheese_df_new).mark_circle().encode(
    x=alt.X('CategoryTypeEn:N', title='Categories of Cheese'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Types of categories of cheese vs Fat Level of the Cheese',
    width=400
)


Category_vs_Fat_plot
```

[112]: alt.Chart(…)

[113]:
```
# Creating EDA to understand the data
# Visualizing "ManufacturerProvCode" vs target variable column "FatLevel" using␣
↪Altair
ManufacturerProv_vs_Fat_plot = alt.Chart(cheese_df_new).mark_circle().encode(
    x=alt.X('ManufacturerProvCode:N', title='Manufacturer Province Code'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Manufacturer Province Code vs Fat Level of the Cheese',
    width=400
)

ManufacturerProv_vs_Fat_plot
```

[113]: alt.Chart(…)

[114]:
```
# Creating EDA to understand the data
# Visualizing "ManufacturingTypeEn" vs target variable column "FatLevel" using␣
↪Altair
ManufactureType_vs_Fat_plot = alt.Chart(cheese_df_new).mark_circle().encode(
    x=alt.X('ManufacturingTypeEn', title='Manufacture Type'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Manufacturing type vs Fat Level of the Cheese',
    width=400
)

ManufactureType_vs_Fat_plot
```

[114]: alt.Chart(…)

[115]:
```
# Creating EDA to understand the data
# Visualizing "MilkTypeEn" vs target variable column "FatLevel" using Altair
MT_vs_Fat_plot = alt.Chart(cheese_df_new).mark_line().encode(
    x=alt.X('MilkTypeEn:N', title='Milk Type'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Milk Type of Cheese vs Fat Levels',
    width=400
)

MT_vs_Fat_plot
```

[115]: alt.Chart(…)

[116]:
```
# Creating EDA to understand the data
# Visualizing "MilkTreatmentTypeEn" vs target variable column "FatLevel" using␣
↪Altair
MTT_vs_Fat_plot = alt.Chart(cheese_df_new).mark_circle().encode(
    x=alt.X('MilkTreatmentTypeEn:N', title='Milk Treatment Type'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Milk Treatment Type of Cheese vs Fat Levels',
    width=400
)

MTT_vs_Fat_plot
```

[116]: alt.Chart(…)

[117]:
```
# Creating EDA to understand the data
# Visualizing "RindTypeEn" vs target variable column "FatLevel" using Altair
RT_vs_Fat_plot = alt.Chart(cheese_df_new).mark_circle().encode(
    x=alt.X('RindTypeEn:N', title='Cheese Rind Type'),
    y=alt.Y('count()', title='Count of Records'),
    color='FatLevel:N'
).properties(
    title='Cheese Rind Type vs Fat Levels',
    width=400
)

RT_vs_Fat_plot
```

[117]: alt.Chart(…)

[118]:
```
# Question 1. Discuss any challenges or peculiarities of the dataset.
# Exploring each sub categorical data values in all the categorical columns
MPV_values = cheese_df_new["ManufacturerProvCode"].value_counts(dropna=False)
print("1. Count of values in ManufacturerProvCode: \n", MPV_values)

MPT_values = cheese_df_new["ManufacturingTypeEn"].value_counts(dropna=False)
print("2. Count of values in ManufacturingTypeEn: \n", MPT_values)

CT_values = cheese_df_new["CategoryTypeEn"].value_counts(dropna=False)
print("3. Count of values in CategoryTypeEn: \n", CT_values)

MT_values = cheese_df_new["MilkTypeEn"].value_counts(dropna=False)
print("4. Count of values in MilkTypeEn: \n", MT_values)
```

```python
MTT_values = cheese_df_new["MilkTreatmentTypeEn"].value_counts(dropna=False)
print("5. Count of values in MilkTreatmentTypeEn: \n", MTT_values)

RT_values = cheese_df_new["RindTypeEn"].value_counts(dropna=False)
print("6. Count of values in RindTypeEn: \n", RT_values)
```

```
1. Count of values in ManufacturerProvCode:
 QC    796
ON    115
BC     65
NB     27
AB     13
MB     11
NS     10
NL      2
PE      2
SK      1
Name: ManufacturerProvCode, dtype: int64
2. Count of values in ManufacturingTypeEn:
 Industrial    455
Artisan       367
Farmstead     220
Name: ManufacturingTypeEn, dtype: int64
3. Count of values in CategoryTypeEn:
 Firm Cheese         349
Soft Cheese         267
Semi-soft Cheese    227
Fresh Cheese        119
Hard Cheese          32
Veined Cheeses       25
NaN                  23
Name: CategoryTypeEn, dtype: int64
4. Count of values in MilkTypeEn:
 Cow                   743
Goat                  214
Ewe                    62
Cow and Goat           13
Ewe and Cow             4
Ewe and Goat            2
Buffalo Cow             2
NaN                     1
Cow, Goat and Ewe       1
Name: MilkTypeEn, dtype: int64
5. Count of values in MilkTreatmentTypeEn:
 Pasteurized    800
Raw Milk       115
NaN             65
```

```
Thermised       62
Name: MilkTreatmentTypeEn, dtype: int64
6. Count of values in RindTypeEn:
 No Rind         404
NaN             321
Bloomy Rind     164
Washed Rind     148
Brushed Rind      5
Name: RindTypeEn, dtype: int64
```

[119]:
```
imputer = SimpleImputer(strategy="most_frequent")
cheese_df_new["RindTypeEn"] = imputer.
 ↪fit_transform(cheese_df_new[["RindTypeEn"]])
cheese_df_new["ManufacturingTypeEn"] = imputer.
 ↪fit_transform(cheese_df_new[["ManufacturingTypeEn"]])
cheese_df_new["CategoryTypeEn"] = imputer.
 ↪fit_transform(cheese_df_new[["CategoryTypeEn"]])
cheese_df_new["MilkTreatmentTypeEn"] = imputer.
 ↪fit_transform(cheese_df_new[["MilkTreatmentTypeEn"]])
cheese_df_new["MoisturePercent"].fillna(0, inplace=True)
```

[120]:
```
# Creating a box plot for numerical column "MoisturePercent" to check for any␣
 ↪outliers in the whiskers of the plot
MP_plot_num = alt.Chart(cheese_df_new).mark_boxplot().encode(
    y='MoisturePercent:Q'
).properties(
    title='Boxplot for Moisture Percent column to check for outliers',
    width=200,
    height=300
)

MP_plot_num
```

[120]: alt.Chart(…)

[121]:
```
# Calculate IQR(Interquartile range) for getting outliers
Q1 = cheese_df_new['MoisturePercent'].quantile(0.25)
Q3 = cheese_df_new['MoisturePercent'].quantile(0.75)
IQR = Q3 - Q1

# Define a threshold for outliers
threshold = 1.5

# Identify and display outliers in "MoisturePercent" column
outliers = cheese_df_new[(cheese_df_new['MoisturePercent'] < Q1 - threshold *␣
 ↪IQR) | (cheese_df_new['MoisturePercent'] > Q3 + threshold * IQR)]
```

```
print("Outliers of Moisture Percent column : ", outliers["MoisturePercent"].
↪count())
```

```
Outliers of Moisture Percent column :  42
```

[122]:
```
# "Tried" to create a bell curve using density plot for numerical column␣
↪"MoisturePercent" to see the distribution of data - skewed or not?

MP_density_plot = alt.Chart(cheese_df_new).transform_density(
    'MoisturePercent',
    as_=['MoisturePercent', 'density'],
).mark_line().encode(
    alt.X('MoisturePercent:Q', title='Moisture Percent'),
    alt.Y('density:Q', title='Density'),
    color=alt.value('blue'),
)

MP_density_plot

# Here its evident from the plot that there is a "normal distribution" of data,␣
↪since it is not left skewed or right skewed.
```

[122]: alt.Chart(…)

[123]:
```
# Checking for class imbalance
unique_classes_FL = cheese_df_new['FatLevel'].unique()
print(unique_classes_FL)
if len(unique_classes_FL) == 2:
    print("It's a binary classification problem.")
else:
    print("It's not a binary classification problem.")
```

```
['lower fat' 'higher fat']
It's a binary classification problem.
```

[124]:
```
# Plotting the class imabalance using Altair
class_counts = pd.DataFrame(cheese_df_new['FatLevel'].value_counts()).
↪reset_index()
class_counts.columns = ['FatLevel', 'Count']
imbalance_chart = alt.Chart(class_counts).mark_bar().encode(
    x='FatLevel:N',
    y='Count:Q',
    color='FatLevel:N'
).properties(
    title='Class Imbalance of FatLevel column',
    width=400
)
```

```
imbalance_chart
```

[124]: alt.Chart(…)

Preprocessing: 1. Take the necessary steps to clean and preprocess the data. 2. Identify different types of features from the dataset and define a column transformer to carry out the necessary preprocessing. 3. Briefly justify your choices : My choice to use categorical and numerical columns like "CategoryTypeEn", "MilkTreatmentTypeEn", "RindTypeEn" and "MoisturePercent" signify that these are great features for classifying fat levels in cheese.

[125]:
```
numeric_features = ["MoisturePercent"]
categorical_features = ["CategoryTypeEn", "MilkTreatmentTypeEn", "RindTypeEn"]

numeric_transformer = Pipeline(
    steps=[("imputer", SimpleImputer(strategy="mean")),
           ("scaler", StandardScaler())]
)

categorical_transformer = Pipeline(
    steps=[("imputer", SimpleImputer(strategy="most_frequent")),
           ("onehot", OneHotEncoder(handle_unknown="ignore"))]
)

col_transformer = ColumnTransformer(
    transformers=[
        ("numeric", numeric_transformer, numeric_features),
        ("categorical", categorical_transformer, categorical_features)
    ],
    remainder='passthrough'
)

col_transformer.fit(cheese_df_new)
cheese_df_new.head()
cheese_df_new.info()
cheese_df_new.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1042 entries, 228 to 2391
Data columns (total 9 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   ManufacturerProvCode  1042 non-null   object
 1   ManufacturingTypeEn   1042 non-null   object
 2   MoisturePercent       1042 non-null   float64
 3   Organic               1042 non-null   int64
 4   CategoryTypeEn        1042 non-null   object
 5   MilkTypeEn            1041 non-null   object
```

```
 6   MilkTreatmentTypeEn   1042 non-null   object
 7   RindTypeEn            1042 non-null   object
 8   FatLevel              1042 non-null   object
dtypes: float64(1), int64(1), object(7)
memory usage: 81.4+ KB
```

[125]:
```
ManufacturerProvCode    0
ManufacturingTypeEn     0
MoisturePercent         0
Organic                 0
CategoryTypeEn          0
MilkTypeEn              1
MilkTreatmentTypeEn     0
RindTypeEn              0
FatLevel                0
dtype: int64
```

[126]:
```
# Applying one hot encoding
#onehot_cols = col_transformer.named_transformers_["categorical"].
↪named_steps["onehot"].get_feature_names(categorical_features)
#onehot_cols

#columns = numeric_features + list(onehot_cols)
#columns
cheese_df_new_category = cheese_df_new.drop(['MoisturePercent', 'Organic',␣
↪'FatLevel'], axis=1)# X_train_category = has only categorical values
ohe = OneHotEncoder(sparse=False, dtype='int')
ohe.fit(cheese_df_new_category);
cheese_df_ohe = ohe.transform(cheese_df_new_category)

cheese_df_ohe # one hot encoded categorical values

ohe_df = pd.DataFrame(
    data=cheese_df_ohe,
    columns=ohe.get_feature_names(['ManufacturerProvCode',␣
↪'ManufacturingTypeEn', 'CategoryTypeEn', 'MilkTypeEn',␣
↪'MilkTreatmentTypeEn', 'RindTypeEn']),
    index=cheese_df_new.index,
)
ohe_df
```

[126]:
```
          ManufacturerProvCode_AB  ManufacturerProvCode_BC  \
CheeseId
228                             0                        0
242                             0                        0
301                             0                        0
303                             0                        0
```

```
319                              0                         0
…                              …                         …
2387                             0                         0
2388                             1                         0
2389                             0                         0
2390                             0                         0
2391                             1                         0

          ManufacturerProvCode_MB  ManufacturerProvCode_NB  \
CheeseId
228                              0                         1
242                              0                         1
301                              0                         0
303                              0                         1
319                              0                         1
…                              …                         …
2387                             0                         0
2388                             0                         0
2389                             0                         0
2390                             0                         0
2391                             0                         0

          ManufacturerProvCode_NL  ManufacturerProvCode_NS  \
CheeseId
228                              0                         0
242                              0                         0
301                              0                         0
303                              0                         0
319                              0                         0
…                              …                         …
2387                             0                         1
2388                             0                         0
2389                             0                         1
2390                             0                         1
2391                             0                         0

          ManufacturerProvCode_ON  ManufacturerProvCode_PE  \
CheeseId
228                              0                         0
242                              0                         0
301                              1                         0
303                              0                         0
319                              0                         0
…                              …                         …
2387                             0                         0
2388                             0                         0
2389                             0                         0
```

```
2390                        0                        0
2391                        0                        0

          ManufacturerProvCode_QC  ManufacturerProvCode_SK  …  \
CheeseId                                                    …
228                             0                        0  …
242                             0                        0  …
301                             0                        0  …
303                             0                        0  …
319                             0                        0  …
…                             …                        …  …
2387                            0                        0  …
2388                            0                        0  …
2389                            0                        0  …
2390                            0                        0  …
2391                            0                        0  …

          MilkTypeEn_Ewe and Goat  MilkTypeEn_Goat  MilkTypeEn_nan  \
CheeseId
228                             0                0               0
242                             0                0               0
301                             0                0               0
303                             0                0               0
319                             0                0               0
…                             …              …             …
2387                            0                0               0
2388                            0                0               0
2389                            0                0               0
2390                            0                0               0
2391                            0                0               0

          MilkTreatmentTypeEn_Pasteurized  MilkTreatmentTypeEn_Raw Milk  \
CheeseId
228                                     0                             1
242                                     0                             1
301                                     1                             0
303                                     0                             1
319                                     0                             1
…                                     …                           …
2387                                    1                             0
2388                                    1                             0
2389                                    0                             0
2390                                    0                             0
2391                                    1                             0

          MilkTreatmentTypeEn_Thermised  RindTypeEn_Bloomy Rind  \
CheeseId
```

```
228                                        0                       0
242                                        0                       0
301                                        0                       0
303                                        0                       0
319                                        0                       0
…                                      …                       …
2387                                       0                       0
2388                                       0                       0
2389                                       1                       0
2390                                       1                       0
2391                                       0                       0

          RindTypeEn_Brushed Rind  RindTypeEn_No Rind  RindTypeEn_Washed Rind
CheeseId
228                             0                   0                       1
242                             0                   0                       1
301                             0                   1                       0
303                             0                   1                       0
319                             0                   0                       1
…                           …                 …                     …
2387                            0                   1                       0
2388                            0                   1                       0
2389                            0                   1                       0
2390                            0                   0                       1
2391                            0                   1                       0

[1042 rows x 35 columns]
```

[127]:

```
# Concating one hot encoded df with the numerical df
cheese_df_new_concat = pd.concat([ohe_df, cheese_df_new['MoisturePercent']],␣
 ↪axis=1)
cheese_df_new_concat2 = pd.concat([cheese_df_new_concat,␣
 ↪cheese_df_new['Organic']], axis=1)
cheese_df_new_concat3 = pd.concat([cheese_df_new_concat2,␣
 ↪cheese_df_new['FatLevel']], axis=1)
cheese_df_new_concat3
```

[127]:

```
          ManufacturerProvCode_AB  ManufacturerProvCode_BC  \
CheeseId
228                             0                        0
242                             0                        0
301                             0                        0
303                             0                        0
319                             0                        0
…                           …                      …
2387                            0                        0
2388                            1                        0
```

```
2389                          0                          0
2390                          0                          0
2391                          1                          0

          ManufacturerProvCode_MB  ManufacturerProvCode_NB  \
CheeseId
228                             0                        1
242                             0                        1
301                             0                        0
303                             0                        1
319                             0                        1
…                             …                        …
2387                            0                        0
2388                            0                        0
2389                            0                        0
2390                            0                        0
2391                            0                        0

          ManufacturerProvCode_NL  ManufacturerProvCode_NS  \
CheeseId
228                             0                        0
242                             0                        0
301                             0                        0
303                             0                        0
319                             0                        0
…                             …                        …
2387                            0                        1
2388                            0                        0
2389                            0                        1
2390                            0                        1
2391                            0                        0

          ManufacturerProvCode_ON  ManufacturerProvCode_PE  \
CheeseId
228                             0                        0
242                             0                        0
301                             1                        0
303                             0                        0
319                             0                        0
…                             …                        …
2387                            0                        0
2388                            0                        0
2389                            0                        0
2390                            0                        0
2391                            0                        0

          ManufacturerProvCode_QC  ManufacturerProvCode_SK  …  \
```

```
CheeseId                                                  …
228                            0                        0  …
242                            0                        0  …
301                            0                        0  …
303                            0                        0  …
319                            0                        0  …
…                            …                        …  …
2387                           0                        0  …
2388                           0                        0  …
2389                           0                        0  …
2390                           0                        0  …
2391                           0                        0  …

          MilkTreatmentTypeEn_Pasteurized  MilkTreatmentTypeEn_Raw Milk  \
CheeseId
228                                     0                             1
242                                     0                             1
301                                     1                             0
303                                     0                             1
319                                     0                             1
…                                     …                           …
2387                                    1                             0
2388                                    1                             0
2389                                    0                             0
2390                                    0                             0
2391                                    1                             0

          MilkTreatmentTypeEn_Thermised  RindTypeEn_Bloomy Rind  \
CheeseId
228                                   0                       0
242                                   0                       0
301                                   0                       0
303                                   0                       0
319                                   0                       0
…                                   …                     …
2387                                  0                       0
2388                                  0                       0
2389                                  1                       0
2390                                  1                       0
2391                                  0                       0

          RindTypeEn_Brushed Rind  RindTypeEn_No Rind  RindTypeEn_Washed Rind  \
CheeseId
228                             0                   0                       1
242                             0                   0                       1
301                             0                   1                       0
303                             0                   1                       0
```

```
319                              0                   0                        1
...                          ...                 ...                     ...
2387                             0                   1                        0
2388                             0                   1                        0
2389                             0                   1                        0
2390                             0                   0                        1
2391                             0                   1                        0

          MoisturePercent  Organic    FatLevel
CheeseId
228                  47.0        0   lower fat
242                  47.9        0   lower fat
301                  54.0        0   lower fat
303                  47.0        0   lower fat
319                  49.4        1   lower fat
...                   ...      ...         ...
2387                 37.0        1  higher fat
2388                 46.0        0   lower fat
2389                 40.0        0  higher fat
2390                 34.0        0  higher fat
2391                 31.5        0  higher fat

[1042 rows x 38 columns]
```

[128]:

```python
# Defining X and y from the dataset
X = cheese_df_new_concat3.drop("FatLevel", axis=1)
y = cheese_df_new_concat3["FatLevel"]
# Splitting the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

Methods & Results : Code Begin by training a baseline model. Proceed to train a basic linear model. Explore and select additional suitable supervised machine learning models appropriate for the problem. Conduct feature engineering (creating new features relevant for the prediction task) and feature selection (if applicable) and hyperparameter optimization for one or two promising models: Did Grid Search Hyperparameter tuning for a promising model KNN with best accuracy. Choose the final model or models based on your chosen evaluation criteria and report the final results on the test set with clarity and detail. Include results for each chosen metric. In case of a classification problem, show confusion matrix and interpret true positives, true negatives, false positives, and false negatives.
Writing Explain the rationale behind your feature engineering and feature selection techniques used to choose relevant features: I think the chosen columns are best choices to predict fat levels in the cheese. Explain the rationale behind the choice of classification algorithms/models: All the models chosen here are from baseline to advanced classification models as they provide better accuracy for the classification. If multiple models were evaluated, provide a concise comparison of their performance: KNN performs best, Decision Tree Classifier has best accuracy at max_depth=9 value of 0.865. Discuss why the chosen model outperformed others or met the project's goals better:

KNN does not take much time to fit and train and also has best accuaracy out of all models, so I applied Grid Search optimization to further choose best parameters and better its F1 score

[132]:
```
# Training baseline model
dum_clf = DummyClassifier(strategy='most_frequent')
dum_clf.fit(X_train, y_train)
y_pred_dum = dum_clf.predict(X_test)
dum_clf.score(X_test, y_test)

print("Classification Report:")
print(classification_report(y_test, y_pred_dum))

accuracy = accuracy_score(y_test, y_pred_dum)
print(accuracy)

conf_matrix_dum = confusion_matrix(y_test, y_pred_dum)
print('Confusion Matrix: \n', conf_matrix_dum)
```

```
Classification Report:
              precision    recall  f1-score   support

  higher fat       0.00      0.00      0.00        68
   lower fat       0.67      1.00      0.81       141

    accuracy                           0.67       209
   macro avg       0.34      0.50      0.40       209
weighted avg       0.46      0.67      0.54       209

0.6746411483253588
Confusion Matrix:
 [[  0  68]
 [  0 141]]

/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1248:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1248:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1248:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[133]:
```
# Training decision tree model
dt_clf = DecisionTreeClassifier(max_depth=9) # tuned HP max_depth = 9 is the␣
↪best accuracy value for this model, accuracy is : 0.856.
dt_clf.fit(X_train, y_train)
y_pred_dt = dt_clf.predict(X_test)
dt_clf.score(X_test, y_test)

print("Classification Report:")
print(classification_report(y_test, y_pred_dt))

accuracy = accuracy_score(y_test, y_pred_dt)
print(accuracy)

conf_matrix_dt = confusion_matrix(y_test, y_pred_dt)
print('Confusion Matrix: \n', conf_matrix_dt)
```

```
Classification Report:
              precision    recall  f1-score   support

  higher fat       0.76      0.79      0.78        68
   lower fat       0.90      0.88      0.89       141

    accuracy                           0.85       209
   macro avg       0.83      0.84      0.83       209
weighted avg       0.85      0.85      0.85       209

0.8516746411483254
Confusion Matrix:
 [[ 54  14]
 [ 17 124]]
```

[135]:
```
# Training random forest model
rdf_clf = RandomForestClassifier(n_estimators=1000)
rdf_clf.fit(X_train, y_train)
y_pred_rdf = rdf_clf.predict(X_test)
rdf_clf.score(X_test, y_test)

print("Classification Report:")
print(classification_report(y_test, y_pred_rdf))

accuracy = accuracy_score(y_test, y_pred_rdf)
print(accuracy)

conf_matrix_rdf = confusion_matrix(y_test, y_pred_rdf)
print('Confusion Matrix: \n', conf_matrix_rdf)
```

```
Classification Report:
```

```
              precision    recall  f1-score   support

  higher fat       0.79      0.68      0.73        68
   lower fat       0.85      0.91      0.88       141

    accuracy                           0.84       209
   macro avg       0.82      0.80      0.81       209
weighted avg       0.83      0.84      0.83       209

0.8373205741626795
Confusion Matrix:
 [[ 46  22]
 [ 12 129]]
```

[141]:

```python
# Training knn model
knn_clf = KNeighborsClassifier(n_neighbors=5)
knn_clf.fit(X_train, y_train)
y_pred_knn = knn_clf.predict(X_test)
knn_clf.score(X_test, y_test)

# Hyperparameter Tuning and Threshold Optimization
# Define the parameter grid for hyperparameter tuning (you can customize this␣
↪based on your needs)
param_grid = {'n_neighbors': [3, 5, 7, 9, 11], 'weights': ['uniform',␣
↪'distance']}
# Perform hyperparameter tuning with GridSearchCV
grid_search = GridSearchCV(knn_clf, param_grid)
grid_search.fit(X_train, y_train)

# Get the best hyperparameters from the grid search
best_n_neighbors = grid_search.best_params_['n_neighbors']
best_weights = grid_search.best_params_['weights']
print(best_n_neighbors)
print(best_weights)

print("Classification Report:")
print(classification_report(y_test, y_pred_knn))

accuracy = accuracy_score(y_test, y_pred_knn)
print(accuracy)

conf_matrix_knn = confusion_matrix(y_test, y_pred_knn)
print('Confusion Matrix: \n', conf_matrix_knn)
```

```
5
distance
Classification Report:
              precision    recall  f1-score   support
```

```
  higher fat       0.83      0.71      0.76        68
   lower fat       0.87      0.93      0.90       141

    accuracy                           0.86       209
   macro avg       0.85      0.82      0.83       209
weighted avg       0.85      0.86      0.85       209

0.8564593301435407
Confusion Matrix:
 [[ 48  20]
 [ 10 131]]
```

[137]:
```python
# Training SVM model
svm_clf = SVC(kernel='rbf', )
svm_clf.fit(X_train, y_train)
y_pred_svm = svm_clf.predict(X_test)
svm_clf.score(X_test, y_test)

print("Classification Report:")
print(classification_report(y_test, y_pred_svm))

accuracy = accuracy_score(y_test, y_pred_svm)
print(accuracy)
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
print('Confusion Matrix: \n', conf_matrix_svm)
```

```
Classification Report:
              precision    recall  f1-score   support

  higher fat       0.61      0.69      0.65        68
   lower fat       0.84      0.79      0.81       141

    accuracy                           0.76       209
   macro avg       0.73      0.74      0.73       209
weighted avg       0.77      0.76      0.76       209

0.7559808612440191
Confusion Matrix:
 [[ 47  21]
 [ 30 111]]
```

Discussion: Write concluding remarks : KNN is the best classification model for this project. Interpret results in the context of project's goals: Project goals are acheived and features like MoisturePercent etc. are best choice for features in predicting target variable. Relate the findings back to the initial problem statement. If your model provides feature importance scores, present and discuss them. Explain which features had the most influence on predictions: MoisturePercent Discuss any limitations of the model or the approach taken: Dataset has less values therefore, high accuracy maybe. Discuss potential sources of bias, data quality issues, or other factors that might affect the results: Maybe the best model for this project might not perform better on unseen data. Discuss other ideas that you did not try but could potentially improve the performance/interpretability: Apply more advanced models. References: 1. Dataset were obtain from Kaggle and follows an Open Government Licence (Canada): https://www.kaggle.com/datasets/jenlooper/cheese 2. https://www.kaggle.com/code/melihkanbay/knn-best-parameters-gridsearchcv 3. https://medium.com/@salemortega/k-neighbors-classifier-with-gridsearchcv-basics-3c445ddeb657