Model training

Try to use different models and compare them to see which one is suitable

In [1]:

```
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split, StratifiedKFold, GridSearchCV
from joblib import dump
import numpy as np
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
from sklearn.model_selection import cross_val_score
```

data scaling and splitting

In [2]:

```
df = pd.read_csv("train_v9rqX0R.csv")

df.drop(columns=[ 'Item_Identifier', 'Item_Weight' ,'Item_MRP', 'Item_Fat_Content',
df.dropna(inplace=True)

category_cols = [col for col in df.columns if df[col].dtype == 'O']
encoders = {}
for col in category_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))
    encoders[col] = le


df.head(5)
```

Out[2]:

	Item_Visibility	Item_Type	Outlet_Identifier	Outlet_Location_Type	Outlet_Type
0	0.016047	4	9	0	1
1	0.019278	14	3	2	2
2	0.016760	10	9	0	1
3	0.000000	6	0	2	0
4	0.000000	9	1	2	1

In [3]:

```
for column, encoder in encoders.items():
    print(f"Column: {column}")
    print(f"Mapping: {dict(enumerate(encoder.classes_))}")
```

```
Column: Item_Type
Mapping: {0: 'Baking Goods', 1: 'Breads', 2: 'Breakfast', 3: 'Canned', 4: 'Dairy',
5: 'Frozen Foods', 6: 'Fruits and Vegetables', 7: 'Hard Drinks', 8: 'Health and Hygi
ene', 9: 'Household', 10: 'Meat', 11: 'Others', 12: 'Seafood', 13: 'Snack Foods', 1
4: 'Soft Drinks', 15: 'Starchy Foods'}
Column: Outlet_Identifier
Mapping: {0: 'OUT010', 1: 'OUT013', 2: 'OUT017', 3: 'OUT018', 4: 'OUT019', 5: 'OUT02
7', 6: 'OUT035', 7: 'OUT045', 8: 'OUT046', 9: 'OUT049'}
Column: Outlet_Location_Type
Mapping: {0: 'Tier 1', 1: 'Tier 2', 2: 'Tier 3'}
Column: Outlet_Type
Mapping: {0: 'Grocery Store', 1: 'Supermarket Type1', 2: 'Supermarket Type2', 3: 'Su
permarket Type3'}
```

In [4]:

```
X = df.iloc[:, df.columns != 'Outlet_Identifier']
y = df.iloc[:, df.columns == 'Outlet_Identifier']

X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=.3, random_state=50
```

Fine tuning (K nearest neighbor)

In [5]:

```
param_grid = {
    'n_neighbors': np.arange(1, 31),
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan'],
    'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute']
}
```

In [6]:

```
knn = KNeighborsClassifier()
```

In [7]:

```
skf = StratifiedKFold(n_splits=3)

grid_search = GridSearchCV(
    knn,
    param_grid,
    cv=skf,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1,
    error_score='raise'
)
```

In [8]:

```
grid_search.fit(X_train, y_train)

print("\nBest parameters:", grid_search.best_params_)
print("Best cross-validation score: {:.4f}".format(grid_search.best_score_))
```

```
Fitting 3 folds for each of 480 candidates, totalling 1440 fits

Best parameters: {'algorithm': 'brute', 'metric': 'manhattan', 'n_neighbors': 23, 'w
eights': 'uniform'}
Best cross-validation score: 0.6572
```

```
C:\Users\User\anaconda3\Lib\site-packages\numpy\ma\core.py:2820: RuntimeWarning: inv
alid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
C:\Users\User\anaconda3\Lib\site-packages\sklearn\neighbors\_classification.py:238:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Pl
ease change the shape of y to (n_samples,), for example using ravel().
  return self._fit(X, y)
```

In [9]:

```
best_model = grid_search.best_estimator_
train_score = best_model.score(X_train, y_train)
test_score = best_model.score(X_test, y_test)

print("\nTraining score: {:.4f}".format(train_score))
print("Test score: {:.4f}".format(test_score))
```

```
Training score: 0.7083
Test score: 0.6598
```

In [10]:

```
dump(best_model, 'knn_best_model.joblib')
```

Out[10]:

```
['knn_best_model.joblib']
```

Fine Tuning (decision tree)

In [11]:

```
dt_model = DecisionTreeClassifier()
```

In [12]:

```
param_grid = {
    'max_depth': [3, 5, 10, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5],
    'criterion': ['gini', 'entropy']
}

# Grid search
grid_search = GridSearchCV(estimator=dt_model, param_grid=param_grid, cv=5, scoring
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)
```

```
Best Parameters: {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_s
amples_split': 2}
```

In [13]:

```
best_dt = grid_search.best_estimator_
scores = cross_val_score(best_dt, X_train, y_train, cv=5)
print("Cross-Validation Accuracy:", scores.mean())
```

```
Cross-Validation Accuracy: 0.6657718601389175
```

In [14]:

```
training_acc = best_dt.score(X_train,y_train)
testing_acc = best_dt.score(X_test,y_test)
training_acc,testing_acc
```

Out[14]:

```
(0.6773382500838082, 0.6722721939773172)
```

In [15]:

```
dump(best_dt, 'decision_tree_best_model.pkl')
```

```
Out[15]: ['decision_tree_best_model.pkl']
```

Because decision tree has a better accuracy and didn't overfitting, I decided to use decision tree model