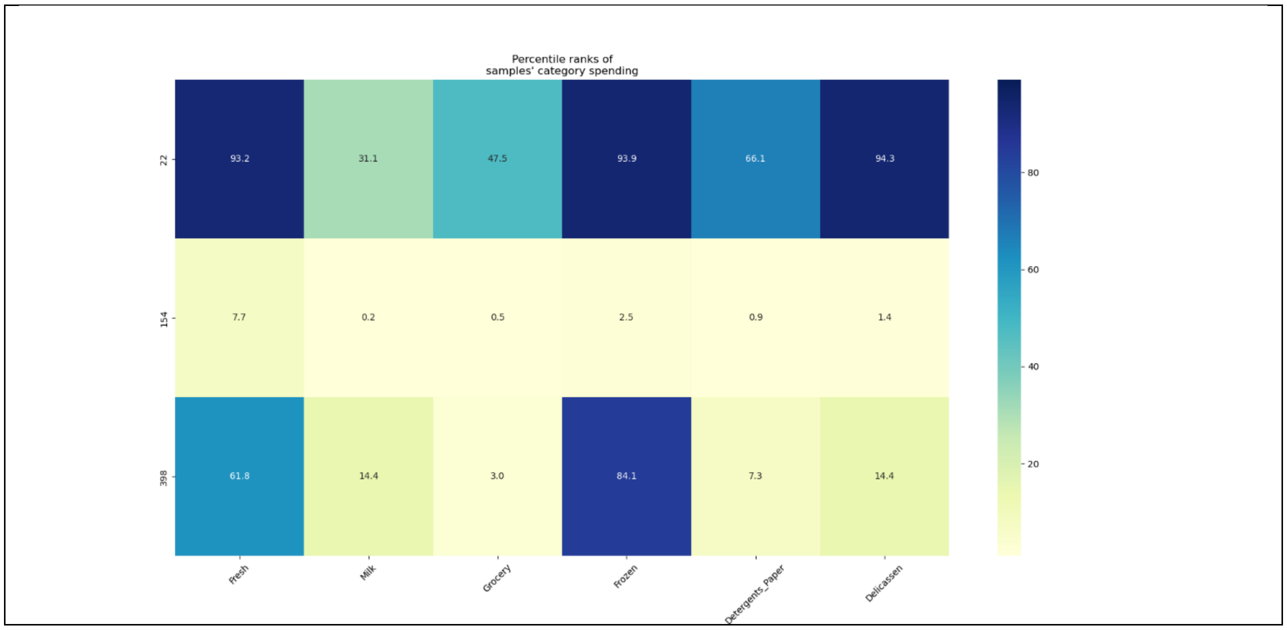
Unsupervised Learning

Data set: Wholesale customers Data Set

The data set comes from UCI Machine Learning Repository. The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units on diverse product categories. The goal here is to use various clustering techniques to segment customers. Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Thus, there is no outcome to be predicted, and the algorithm just tries to find patterns in the data.



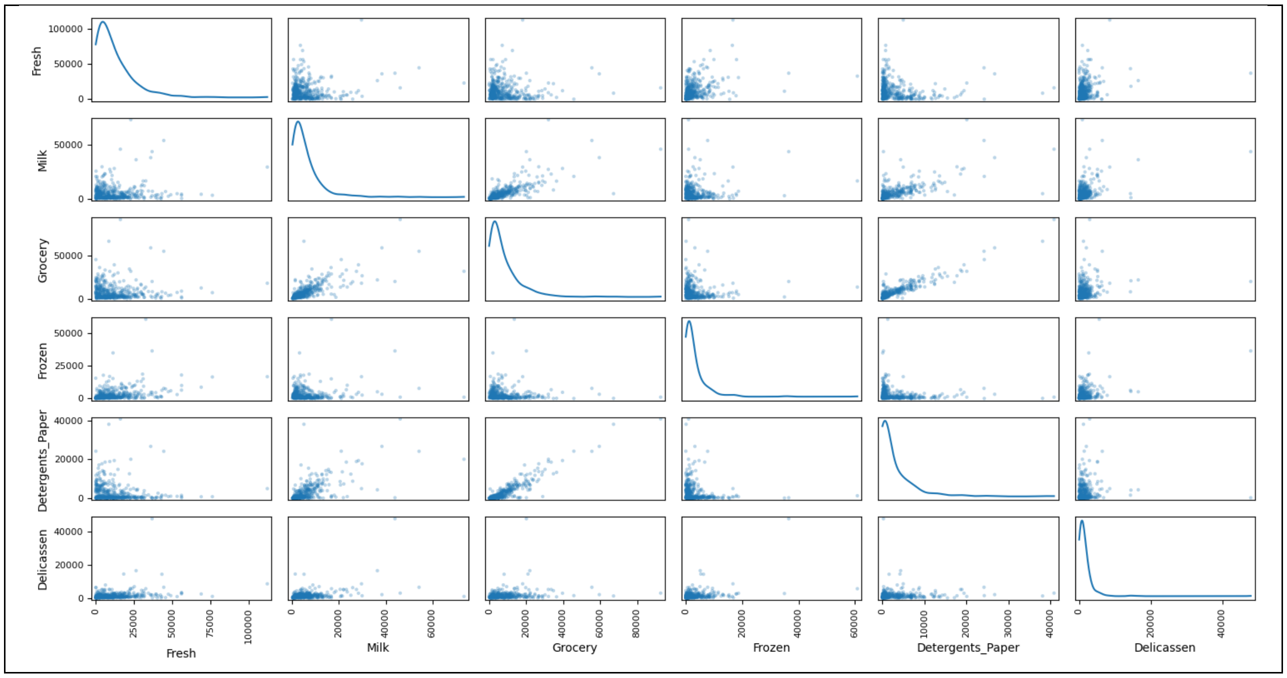
Samples: - 0: This customer ranks above the 90th percentile for annual spending amounts in Fresh, Frozen, and the Delicatessen categories. These features along with above average spending for detergents paper could lead us to believe this customer is a market. Markets generally put an emphasis on having a large variety of fresh foods available and often contain a delicatessen or deli.¶

1: On the opposite side of the spectrum, this customer ranks in the bottom 10th percentile across all product categories. Its highest-ranking category is 'Fresh' which might suggest it is a small cafe or similar.

2: Our last customer spends a lot in the Fresh and Frozen categories but Moreso in the latter. I would suspect this is a wholesale retailer because of the focus on Fresh and Frozen foods.

One interesting thought to consider is if one (or more) of the six product categories is actually relevant for understanding customer purchasing. That is to say, is it possible to determine whether customers purchasing some amount of one category of products will necessarily purchase some proportional amount of another category of products? We can make this determination quite easily by training a supervised regression learner on a subset of the data with one feature removed, and then score how well that model can predict the removed feature.

As you can see, we attempted to predict Milk using the other features in the dataset and the score ended up being 0.515. At this initial stage we might say that this feature is somewhat difficult to predict because the score is around the halfway point of possible scores. Remember that R^2 goes from 0 to 1. This might indicate that it could be an important feature to consider.

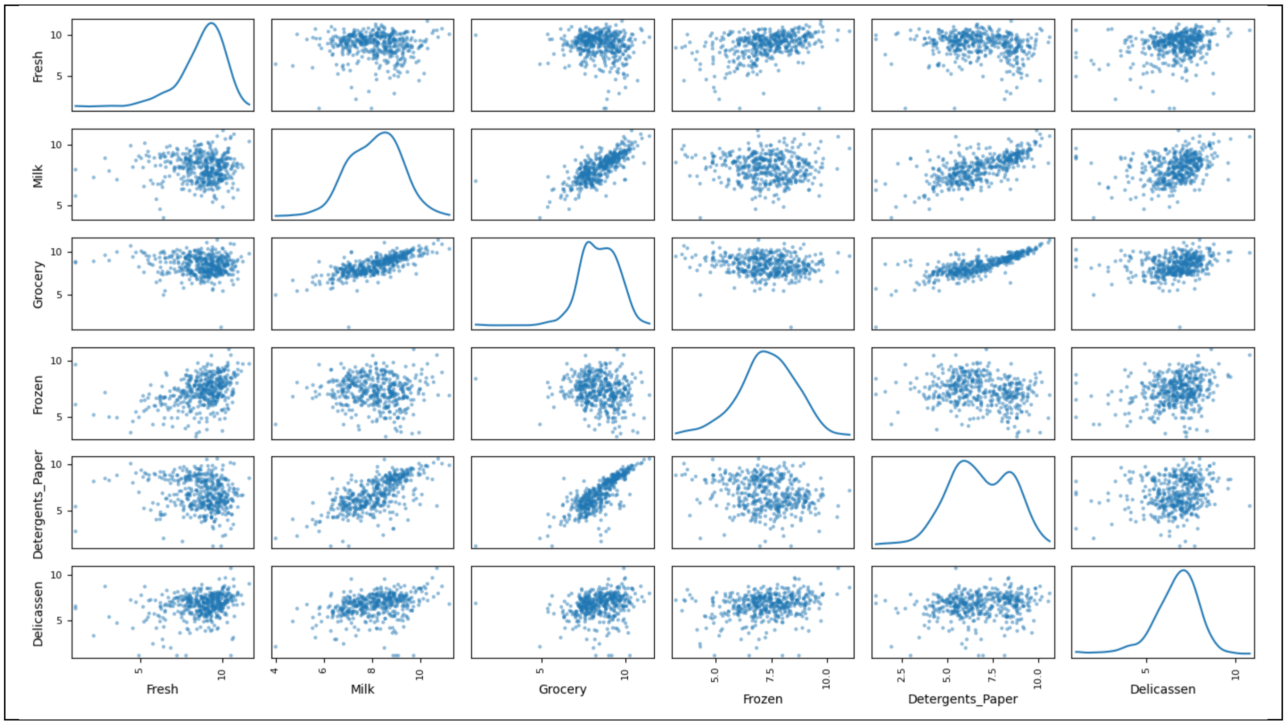


Milk showed some signs of correlation for about half of the features it was compared to which aligns with our earlier prediction. The pair of features with the highest correlation are Detergents_Paper and Grocery which intuitively makes sense as many people shop for both when they go "grocery shopping." One other visible point to note is how many of the points are around 0 for features compared to Delicatessen. The data for all of these features are right-skewed with many points hovering at the origin or near it and long tails.

Data Preprocessing

Feature Scaling:

After scaling the features, the graph shows the data normalization as below:



The distribution of each feature now appears much more normal than before.

Python code for outlier detection, feature transformation and PCA below:

```
import copy, math

import numpy as np

import matplotlib.pyplot as plt

import xlsxwriter

import openpyxl

import pandas as pd

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import os

import csv
```

```
from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import StandardScaler, PolynomialFeatures

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error

import seaborn as sbs


rows=[]

csvpath="E:\\Data"


#csvpath=input("Enter the path of the alarms files: ")

csvfile=os.listdir(csvpath)

for file in csvfile:

    file1 = open(os.path.join(csvpath, file))

    csvreader = csv.reader(file1)

    #concrete_data=pd.read_csv(file1)

    #concrete_data.head()

    for row in csvreader:

        line = str(row)

        line1 = line.rstrip()

        lst = line1.split(",")


data = pd.read_csv("E:\\Data\\Wholesale customers data.csv")

data.drop(labels=(['Channel','Region']),axis=1,inplace=True)


print('Wholesale customers has {} samples with {} features each'.format(*data.shape))
```

```
# Select three indices of your choice you wish to sample from the dataset

indices = [22,154,398]

samples = pd.DataFrame(data.loc[indices], columns=data.keys()).reset_index(drop=True)



pcts = 100. * data.rank(axis=0, pct=True).iloc[indices].round(decimals=3)

# visualize percentiles with heatmap



sns.heatmap(pcts, annot=True, vmin=1, vmax=99, fmt='.1f', cmap='YlGnBu')

plt.title('Percentile ranks of\nsamples\' category spending')

plt.xticks(rotation=45, ha='center');



plt.show()

from sklearn.tree import DecisionTreeRegressor

from sklearn.model_selection import train_test_split



# Remove 1 column

new_data = data.drop('Milk',axis=1)

# Splitting the data into training and testing sets(0.25) using the given feature as the target

X_train, X_test, y_train, y_test = train_test_split(new_data, data['Milk'], test_size=0.25,
random_state=1)

# Create a decision tree regressor and fit it to the training set

regressor =  DecisionTreeRegressor(random_state=1)

regressor.fit(X_train, y_train)

# Report the score of the prediction using the testing set

score = regressor.score(X_test, y_test)
```

```
print("Score = ",score)

pd.plotting.scatter_matrix(data, alpha=0.3,figsize=(15,8),diagonal='kde' )

plt.tight_layout() # To avoid overlapping of plots

plt.show()

# Scale the data using the natural logarithm

log_data = np.log(data.copy())


# Scale the sample data using the natural logarithm

log_samples = np.log(samples)


# Produce a scatter matrix for each pair of newly-transformed features

pd.plotting.scatter_matrix(log_data, alpha=0.5, figsize=(14,8),diagonal='kde')

plt.tight_layout()

plt.show()

# Scale the data using the natural logarithm

log_data = np.log(data.copy())


# Scale the sample data using the natural logarithm

log_samples = np.log(samples)


# Produce a scatter matrix for each pair of newly-transformed features

pd.plotting.scatter_matrix(log_data, alpha=0.5, figsize=(14,8),diagonal='kde')

plt.tight_layout()

# Let's compare the original sample data to the log-transformed sample data

print("Original chosen samples of wholesale customers dataset:")
print(samples)


# Display the log-transformed sample data

print("Log-transformed samples of wholesale customers dataset:")

print(log_samples)

# For each feature find the data points with extreme high or low values

for feature in log_data.keys():

    # Calculate Q1 (25th percentile of the data) for the given feature

    Q1 = np.percentile(log_data, 25)


    # Calculate Q3 (75th percentile of the data) for the given feature

    Q3 = np.percentile(log_data, 75)


    # Use the interquartile range to calculate an outlier step (1.5 times the interquartile range)

    step = (Q3 - Q1) * 1.5


    # Display the outliers

    print("Data points considered outliers for the feature '{}':".format(feature))

    print(log_data[~((log_data[feature] >= Q1 - step) & (log_data[feature] <= Q3 + step))])


    # Select the indices for data points you wish to remove

outliers = [66, 75, 338, 142, 154, 289]


# Remove the outliers, if any were specified

good_data = log_data.drop(log_data.index[outliers]).reset_index(drop=True)
```

```
#Implementation: PCA

from sklearn.decomposition import PCA


# Apply PCA by fitting the good data with the same number of dimensions as features

pca = PCA(n_components=6)

pca.fit(good_data)


# Transform log_samples using the PCA fit above

pca_samples = pca.transform(log_samples)

print(pca.components_)

print(pca.explained_variance_)

print(pca_samples)
```