Professional Masters in Applied Statistics and Data Science (ASDS)
Course Title: Introduction to Data Science with Python
Course No.: PM-ASDS04 (Section: B)

Assignment-2

Prepared for:
Dr. Rumana Rois
Department of Statistics
Jahangirnagar University

Prepared by:
A. M. Abeid Hassan
ID # 2016
PM-ASDS
2nd Batch
Section-B

Table of Contents

1. Read 'wholesale_customers_data.csv' --2

1.i Illustrate summary statistics---3

**1.ii Calculate the covariance matrix and correlation matrix of the variables.
Interpret the results.---4**

**1.iii Graphically examine and interpret region-wise and channel-wise customers
distribution---10**

**2 Use the unsupervised learning method - for clustering customers into
different groups--- 12**

3 Categorize different customer's types (Potential > 33000, Average <=33000) --------------------4

**3.i Run the Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines
to classify the different categories of customers---15**

3.ii Use K-fold Cross-Validation (k=5) to find the best technique among them-------------------17

**3.iii Find the confusion matrix and ROC curve for the best method. Hence, calculate and
interpret: Predictive value positive and negative, Accuracy, Sensitivity, and Specificity of the
test.---18**

1. Read 'wholesale_customers_data.csv'

Python codes

```
# Reading 'wholesale_customers_data.csv' into DataFrame 'df'
df = pd.read_csv("/home/abeid/Documents/2-pm-asds/04-intro-to-data-science-with-python/
assignment-2/wholesale_customers_data.csv")

# df shape
df.shape
```

Result

(440, 8)

Interpretation

Wholesale customers dataset has 440 rows of data across 8 variables.

Append rows into DataFrame 'df'

Python codes

```
# creating list of data series
datarowsSeries = [pd.Series([1, 3, 39228, 1431, 764, 4510,93,2346], index=df.columns),
            pd.Series([1, 3, 14531, 1548, 343,437,841, 1800], index=df.columns),
            pd.Series([1, 3, 1016, 2016, 2016, 1016, 716, 2116], index=df.columns),
            pd.Series([1, 3, 1036, 2026, 2066, 1046, 1716, 2516], index=df.columns),
            pd.Series([1, 3, 1056, 2036, 2066, 1066, 2716, 2716], index=df.columns),
            pd.Series([1, 3, 1076, 2046, 2066, 1086, 3716, 2916], index=df.columns)]

# appending list of data series with 'df' and assigning it to DataFrame 'df_new'
df_new =  df.append(datarowsSeries, ignore_index=True)

df_new.shape
```

Result

(446, 8)

Interpretation

The new data frame has 446 rows and 8 variables after adding 6 rows.

1.i. Illustrate summary statistics

Python codes

```
# creating DataFrame with categorical variables 'df_newcat'
df_newcat = df_new.drop(["Fresh", "Milk", "Grocery", "Frozen", "Detergents_Paper", "Delicassen"],
axis=1)

# creating DataFrame with numeric variables 'df_newnum' by dropping categorical variables from
'df_new'
df_newnum = df_new.drop(["Channel", "Region"], axis=1)

# Descriptive Statistics for Numeric Variables
df_newnum.describe()

# Summary Measures for Categorical Variables
# mode of categorical variables in 'mode_cat'
mode_cat = df_newcat.mode()

#renaming row label of 'mode_cat' DataFrame
mode_cat.rename(index={0:'mode'}, inplace=True)
mode_cat
```

Result

Summary Statistics for Whole Sale Customers Dataset (Numeric Variables)

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
count	446.000000	446.000000	446.00000	446.00000	446.000000	446.000000
mean	11968.775785	5743.183857	7865.20852	3051.14574	2864.697309	1536.665919
std	12671.077458	7344.626047	9468.04341	4827.68344	4740.010059	2803.184225
min	3.000000	55.000000	3.00000	25.00000	3.000000	3.000000
25%	3089.500000	1537.500000	2115.00000	744.00000	258.500000	411.250000
50%	8413.500000	3607.500000	4725.00000	1498.00000	820.500000	982.500000
75%	16905.500000	7141.000000	10510.25000	3543.50000	3888.500000	1836.250000
max	112151.000000	73498.000000	92780.00000	60869.00000	40827.000000	47943.000000

Mode of Categorical Variables

	Channel	Region
mode	1	3

Interpretation

Channel and Region are categorical variables.
The rest of the 6 variables are ratio.
Annual customer spending is highest on Fresh products followed by Milk and Groceries on average.
On average customers spend the least on Delicassen products.
Most frequent customers are Hotel/Resaurant/Cafe.
Highest number of customers are from 'Other' region.

1.ii Calculate the covariance matrix and correlation matrix of the variables. Interpret the results.

Python Codes

```
# Plotting Scatter Dagrams to Evaluate the Relationship between Variables
sns.set_style('whitegrid');
sns.pairplot(df_newnum)
# plt.title('Figure-3: Scatterplot Matrix')
plt.show()
```

Correlation matrix has been computed for the numeric variables. Channel and Region are nominal variables and hence, their correlation cannot be computed.

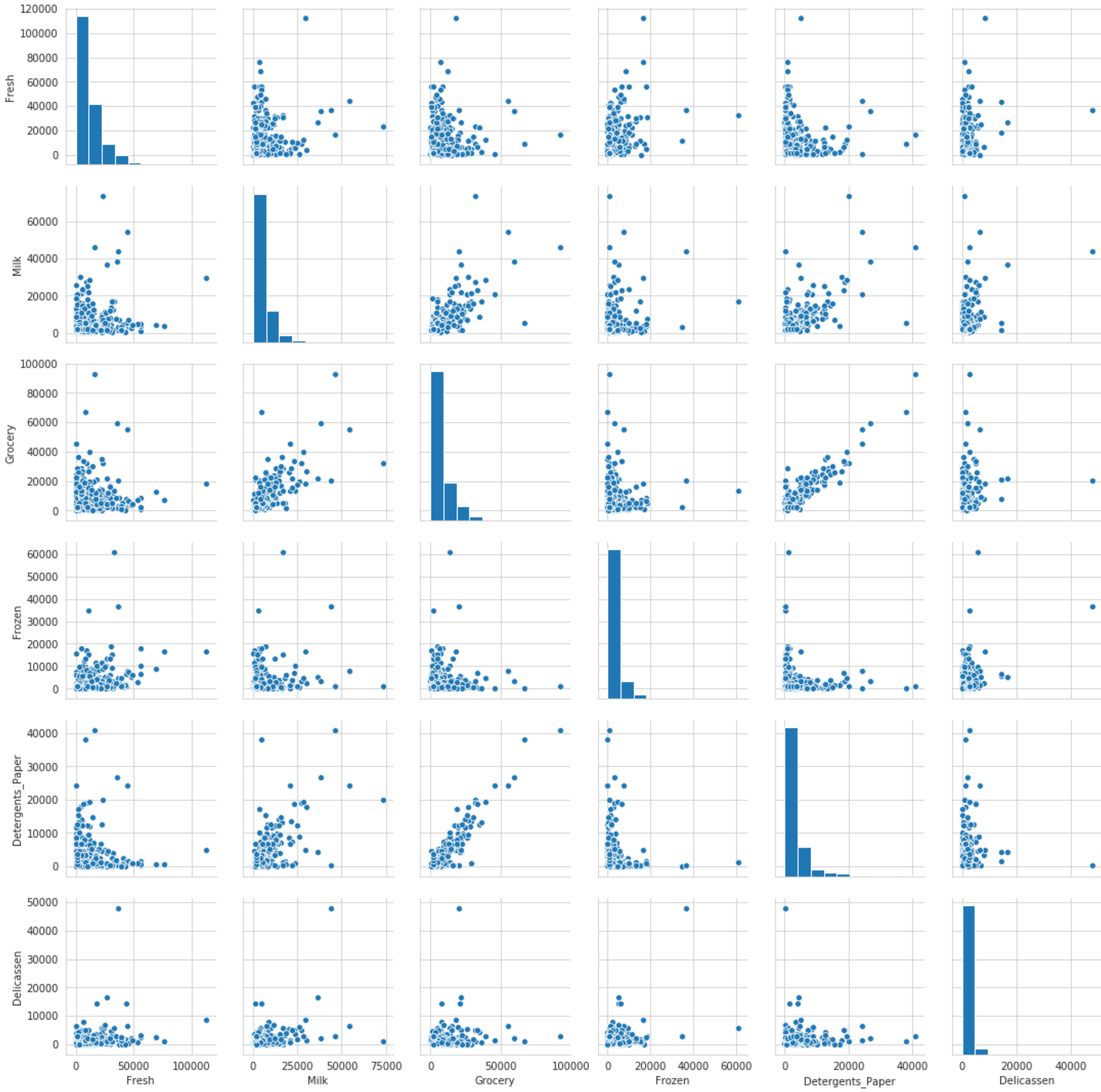
```
# Correlation Matrix for Numeric Variables with Linear Relationship
df_newnum.drop(["Fresh","Frozen", "Delicassen"], axis=1).corr()

# Correlation Test (Spearman's r) for Numeric Variables with Non-Linear Relationship
# Spearman's r for Fresh and Milk
coef, p = sc.spearmanr(df_new.Fresh, df_new.Milk)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Fresh and Grocery
coef, p = sc.spearmanr(df_new.Fresh, df_new.Grocery)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)
```

```
# Spearman's r for Fresh and Frozen
coef, p = sc.spearmanr(df_new.Fresh, df_new.Frozen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Fresh and Detergents_Paper
coef, p = sc.spearmanr(df_new.Fresh, df_new.Detergents_Paper)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Fresh and Delicassen
coef, p = sc.spearmanr(df_new.Fresh, df_new.Delicassen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Milk and Frozen
coef, p = sc.spearmanr(df_new.Milk, df_new.Frozen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f'

# Spearman's r for Milk and Delicassen
coef, p = sc.spearmanr(df_new.Milk, df_new.Delicassen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
```

```
	print('Samples are correlated (reject H0) p=%.3f'

# Spearman's r for Grocery and Frozen
coef, p = sc.spearmanr(df_new.Grocery, df_new.Frozen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Grocery and Delicassen
coef, p = sc.spearmanr(df_new.Grocery, df_new.Delicassen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Frozen and Detergents_Paper
coef, p = sc.spearmanr(df_new.Frozen, df_new.Detergents_Paper)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Frozen and Delicassen
coef, p = sc.spearmanr(df_new.Frozen, df_new.Delicassen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
	print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
	print('Samples are correlated (reject H0) p=%.3f' % p)

# Spearman's r for Detergents_Paper and Delicassen
coef, p = sc.spearmanr(df_new.Detergents_Paper, df_new.Delicassen)
print('Spearmans correlation coefficient: %.3f' % coef)
# interpret the significance
alpha = 0.05
if p > alpha:
```

```
    print('Samples are not correlated (fail to reject H0) p=%.3f' % p)
else:
    print('Samples are correlated (reject H0) p=%.3f' % p)
```

Result:



Scatter Plot Matrix of Wholesale Customers Dataset (Numeric Variables)

	Milk	Grocery	Detergents_Paper
Milk	1.000000	0.729554	0.661894
Grocery	0.729554	1.000000	0.923474
Detergents_Paper	0.661894	0.923474	1.000000

Correlation Matrix of Numeric Variables with Linear Relationship

Spearman's r Test:

Fresh and Milk
Spearmans correlation coefficient: -0.080
Samples are not correlated (fail to reject H0) p=0.092

Fresh and Grocery
Spearmans correlation coefficient: -0.114
Samples are correlated (reject H0) p=0.016

Fresh and Frozen
Spearmans correlation coefficient: 0.386
Samples are correlated (reject H0) p=0.000

Fresh and Detergents_Paper
Spearmans correlation coefficient: -0.206
Samples are correlated (reject H0) p=0.000

Fresh and Delicassen
Spearmans correlation coefficient: 0.225
Samples are correlated (reject H0) p=0.000

Milk and Frozen
Spearmans correlation coefficient: -0.090
Samples are not correlated (fail to reject H0) p=0.057

Milk and Delicassen
Spearmans correlation coefficient: 0.359
Samples are correlated (reject H0) p=0.000

Grocery and Frozen
Spearmans correlation coefficient: -0.162
Samples are correlated (reject H0) p=0.001

Grocery and Delicassen
Spearmans correlation coefficient: 0.282
Samples are correlated (reject H0) p=0.000

Frozen and Detergents_Paper
Spearmans correlation coefficient: -0.212

Samples are correlated (reject H0) p=0.000

Frozen and Delicassen
Spearmans correlation coefficient: 0.226
Samples are correlated (reject H0) p=0.000

Detergents_Paper and Delicassen
Spearmans correlation coefficient: 0.184
Samples are correlated (reject H0) p=0.000

Interpretation:

From Scatter Plot Matrix the following relationships have been identified:
Linear relationship: Milk and Grocery, Milk and Detergents_Paper, Grocery and Detergents_Paper,

Monotonic relationship: None

Non-linear relationship: Fresh and Milk, Fresh and Grocery, Fresh and Frozen, Fresh and Detergents_Paper,
Fresh and Delicassen, Milk and Frozen, Milk and Delicassen, Grocery and Frozen, Grocery and Delicassen, Frozen and Detergents_Paper, Frozen and Delicassen, Detergents_Paper and Delicassen

From the Correlation Matrix the following have been determined:
Strong positive correlation exists between Grocery and Detergents_Paper as indicated by pearson's r of 0.92 which
means that if annual spending on Grocery products increases, then it will also increase on Detergents_Paper products and vice versa.

Pearson's r of 0.73 shows a moderate positive correlation between Milk and Grocery.

Milk and Detergents_Paper also have a moderate positive correlation between them with Pearson's r of 0.66.

From the Spearman's r Correlation Test the following have been determined:
Fresh and Milk are not correlated (p > alpha).
Fresh and Grocery are correlated. They have very weak negative relationship as shown by r of -0.11.
Fresh and Frozen are correlated. They have weak positive relationship (r = 0.39).
Fresh and Detergents_Paper are correlated. They have weak negative relationship (r = 0.21).
Fresh and Delicassen have a weak positive relationship (r = 0.23).
Milk and Frozen are not correlated (p > alpha).
Milk and Delicassen have a weak positive correlation (r = 0.36).
Grocery and Frozen have a very weak negative correlation (r = -0.162).
Grocery and Delicassen have a weak positive correlation (r= 0.28).
Frozen and Detergents_Paper have a weak neagative correlation (r= 0.21).
Frozen and Delicassen have a weak positive correlation (r= 0.23).

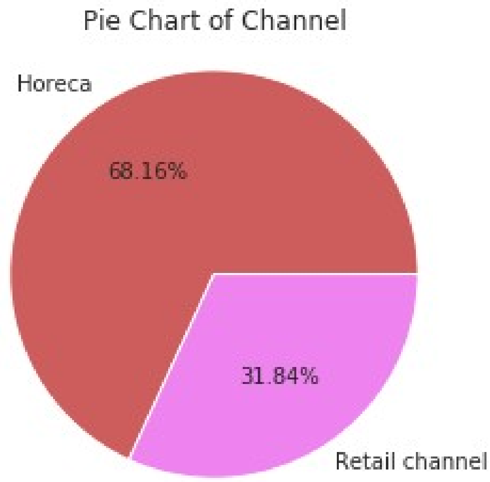
#Detergents_Paper and Delicassen have a very weak positive correlation (r= 0.18).

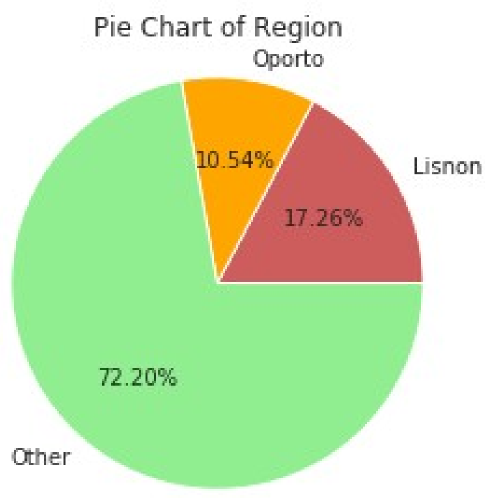
1.iii Graphically examine and interpret region-wise and channel-wise customers distribution

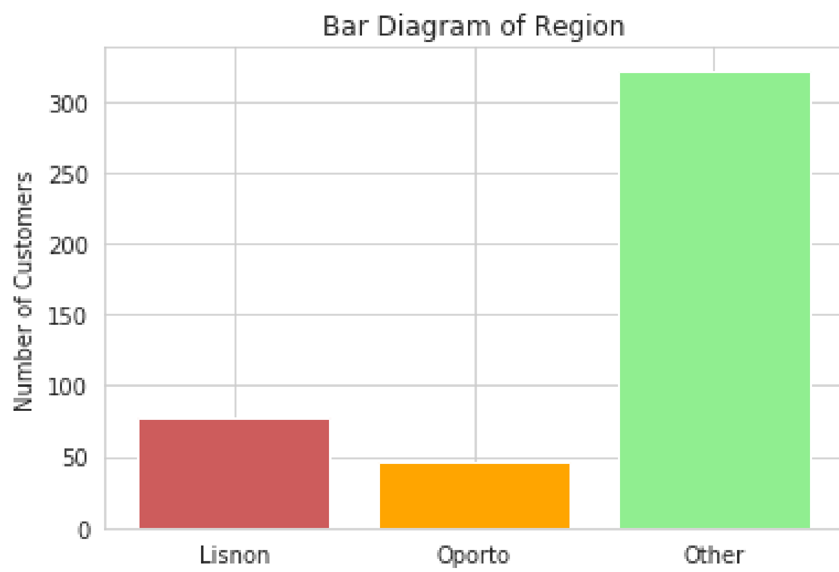
Python Codes

```
# Pie Chart of Channel
# frequency of channel
# df_new["Channel"].value_counts().sort_index()
channel_fre = [304, 142]
labels = ['Horeca', 'Retail channel']
colors = ['indianred', 'violet']
plt.pie(channel_fre,labels=labels, colors=colors, autopct='%.2f%%', radius = 1.1)
plt.title('Pie Chart of Channel')
plt.show()


# Pie Chart of Region
# frequency of region
# df_new["Region"].value_counts().sort_index()
channel_fre = [77, 47, 322]
labels = ['Lisnon', 'Oporto', 'Other']
colors = ['indianred', 'orange', 'lightgreen']
plt.pie(channel_fre,labels=labels, colors=colors, autopct='%.2f%%', radius = 1.1)
plt.title('Pie Chart of Region')
plt.show()


#Bar diagram of Region
objects = ('Lisnon', 'Oporto', 'Other')
color = ['indianred', 'orange', 'lightgreen']
x_pos = np.arange(len(objects))
quality_fre = [77, 47, 322]
plt.bar(x_pos, quality_fre, color=color)
plt.xticks(x_pos, objects)
plt.ylabel('Number of Customers')
plt.title('Bar Diagram of Region')
plt.show()
```

Results


Pie Chart of Channel
Horeca
68.16%
31.84%
Retail channel


Pie Chart of Region
Oporto
10.54%
Lisnon
17.26%
72.20%
Other



Interpretation

From the Pie Chart of Channel it can be illustrated that approaximately 68% of the customers represent Hotel/Restaurant/Cafe, and 32% (approax.) of the customers represents Retail channel. The Pie Chart of Region shows that most customers belong to 'Other' region (72% approax.). Approximately 17% are from Lisnon and 11% from Oporto.

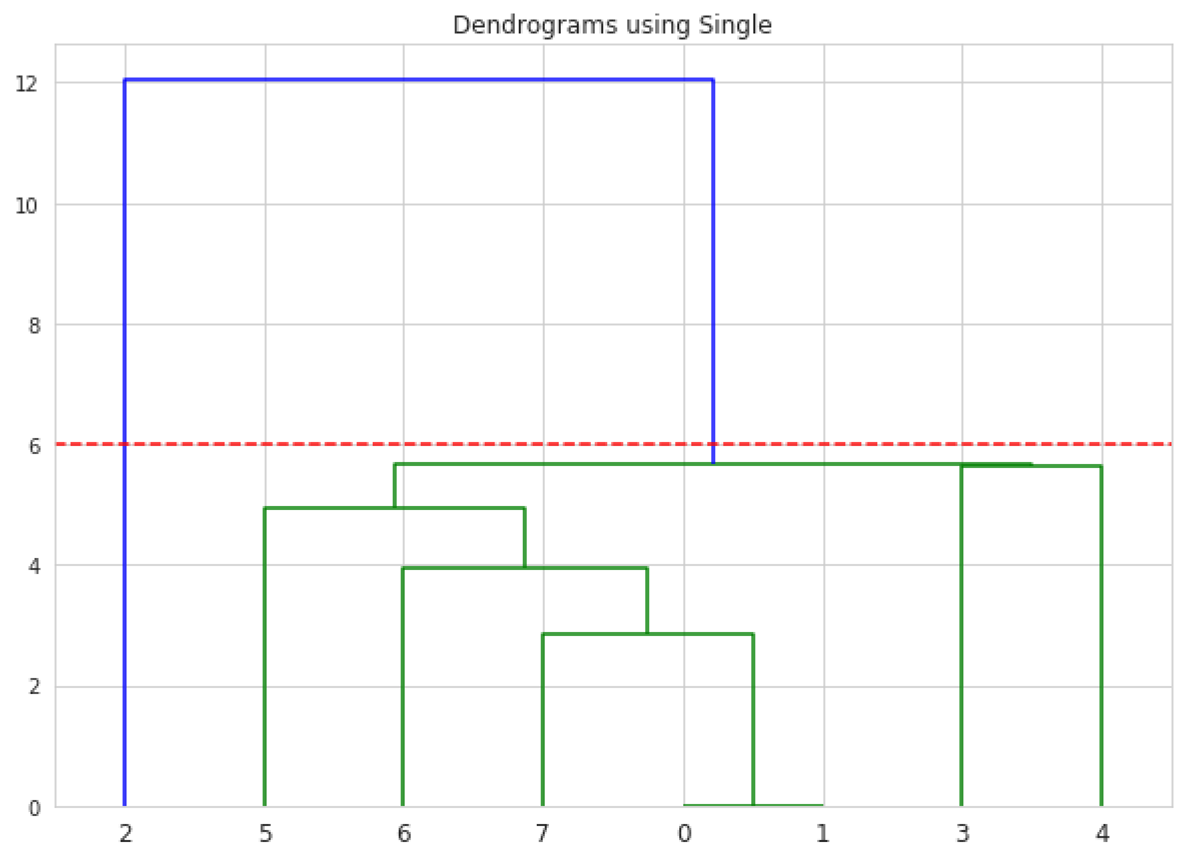
2 Use the unsupervised learning method - for clustering customers into different groups

Python Codes

```
# Clustering

# Clustering data with Dendrogram using Single Linkage

# normalizing the data and bringing all the variables to the same scale
from sklearn.preprocessing import normalize
data_scaled = normalize(df_new)

# creating DataFrame for the scaled data
data_scaled = pd.DataFrame(data_scaled, columns=df_new.columns)
data_scaled.head()

# transposing data_scaled for variable-wise clustering
data_scaled=data_scaled.transpose()
data_scaled.head()
```

```
# plotting dendrogram
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograms using Single")
dend = shc.dendrogram(shc.linkage(data_scaled, method='single'))

# the blue line at x = 2 is the vertical line with the maximum distance. Hence, setting a threshold at the midpoint of the blue line (y = 6).
plt.axhline(y=6, color='r', linestyle='--')
plt.show()


# Hierarchical Agglomerative Clustering with Single Linkage for 2 Clusters
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='single')
cluster.fit_predict(data_scaled)


# Creating a new variable 'GROUP' in the data frame
data_scaled['GROUP']=cluster.fit_predict(data_scaled)
data_scaled['GROUP']
```

Result



Hierarchichal Agglomerative Clustering (Variable-wise):

```
Channel             0
Region              0
Fresh               1
Milk                0
Grocery             0
Frozen              0
Detergents_Paper    0
Delicassen          0
Name: GROUP, dtype: int64
```

Interpretation

In the Dendrogram the threshold (red dotted line) intersects two vertical lines. Thus, our number of clusters is 2.

Exploring the features of the different groups from Hierarchichal Agglomerative Clustering (Variable-wise):
Features in Group-1: Fresh
Feature in Group-0: Channel, Region, Fresh, Milk, Grocery, Frozen, Detergents_Paper, Delicassen

3. Categorize different customer's types (Potential > 33000, Average <=33000)

Python Codes

```
# creating Total_Consumption by summing all Annual spending variables
df_new['Total_Consumption'] = df_new[list(df.columns[[2,3,4,5,6,7]])].sum(axis=1)
df_new.head()

# grouping customers with Total_Consumption > 33,000 as Customer_Type 1 and Total_Consumption <= 33,000 as Customer_Type 0
df_new['Customer_Type']=np.where(df_new['Total_Consumption']<=33000, '0', '1')
df_new.head()
```

Result

Whole Sale Customer Data with Total Consumption & Customer_Type (first 5 rows)

	Channel	**Region**	**Fresh**	**Milk**	**Grocery**	**Frozen**	**Detergents_Paper**	**Delicassen**	**Total_Consumption**	**Customer_Type***
0	2	3	12669	9656	7561	214	2674	1338	34112	1
1	2	3	7057	9810	9568	1762	3293	1776	33266	1
2	2	3	6353	8808	7684	2405	3516	7844	36610	1
3	1	3	13265	1196	4221	6404	507	1788	27381	0

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen	Total_Consumption	Customer_Type*
4	2	3	22615	5410	7198	3915	1777	5185	46100	1

* 0:Average Customers 1:Potential Customers

3.i Run the Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines to classify the different categories of customers

Python Codes

```
# Classifying the Different Categories of Customers Using Logistic Regression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
cols = ['Channel','Region','Fresh','Milk','Grocery','Frozen','Detergents_Paper', 'Delicassen']
X = df_new[cols]
y = df_new['Customer_Type']
logmodel = LogisticRegression()
logmodel.fit(X, y)
print('Coefficients:', logmodel.coef_)
print('Intercept:', logmodel.intercept_)

# Classifying the Different Categories of Customers Using Decision Trees, Random Forest & Support Vector Machines

# Splitting data
cols = ['Channel','Region','Fresh','Milk','Grocery','Frozen','Detergents_Paper', 'Delicassen']
X = df_new[cols]
y = df_new['Customer_Type']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =train_test_split(X, y, test_size=.3)

# DT
from sklearn import tree
DTclf=tree.DecisionTreeClassifier()
DTclf.fit(X_train, y_train)
y_pred= DTclf.predict(X_test)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(classification_report(y_test, y_pred))

# RF
from sklearn.ensemble import RandomForestClassifier
RFclf=RandomForestClassifier(n_estimators=11)
RFclf.fit(X_train, y_train)
y_pred=RFclf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
# SVC
from sklearn.svm import SVC
SVclf = SVC(kernel='poly', degree=4)
#### kernel='linear', Gaussian kernel: kernel = 'rbf', kernel='sigmoid'
SVclf.fit(X_train, y_train)
y_pred=SVclf.predict(X_test)
print(classification_report(y_test, y_pred))
```

Result

Logistic Regression Model

Coefficients: [[-1.51385006e+00 -1.05652097e+00 2.01508240e-04 2.48788122e-04
  2.23217578e-04 1.88344138e-04 3.54990317e-04 3.46784417e-04]]

Intercept: [-2.79984773]

Classifaction report for Decision Tree Model

	precision	recall	f1-score	support
0	0.93	0.87	0.90	79
1	0.83	0.91	0.87	55
micro avg	0.89	0.89	0.89	134
macro avg	0.88	0.89	0.89	134
weighted avg	0.89	0.89	0.89	134

Classifaction report for Random Forest Model

	precision	recall	f1-score	support
0	0.95	0.91	0.93	79
1	0.88	0.93	0.90	55
micro avg	0.92	0.92	0.92	134
macro avg	0.91	0.92	0.92	134
weighted avg	0.92	0.92	0.92	134

Classifaction report for Support Vector Machines Model

	precision	recall	f1-score	support
0	0.99	1.00	0.99	79
1	1.00	0.98	0.99	55
micro avg	0.99	0.99	0.99	134
macro avg	0.99	0.99	0.99	134
weighted avg	0.99	0.99	0.99	134

3.ii Use K-fold Cross-Validation (k=5) to find the best technique among them

Python Codes

```
cols = ['Channel','Region','Fresh','Milk','Grocery','Frozen','Detergents_Paper', 'Delicassen']
X = df_new[cols]
y = df_new['Customer_Type']

DTscores = []
RFscores = []
SVscores = []
LRscores = []

#Logistic Regression
from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()


# DT
from sklearn import tree
DTclf=tree.DecisionTreeClassifier()

#RF
from sklearn.ensemble import RandomForestClassifier
RFclf=RandomForestClassifier(n_estimators=11)

#SVM
from sklearn.svm import SVC
SVclf = SVC(kernel='poly', degree=4)


from sklearn.model_selection import KFold
cv = KFold(n_splits=5, random_state=1, shuffle=True)
for train_index, test_index in cv.split(X):
    print("Train Index: ", train_index)
    print("Test Index: ", test_index, "\n")

    X_train, X_test, y_train, y_test = X.iloc[train_index], X.iloc[test_index], y.iloc[train_index],
y.iloc[test_index]
    ############# For LR ##################
    logmodel.fit(X, y)
    LRscores.append(logmodel.score(X_test, y_test))
    ############# for DT #################
    DTclf.fit(X_train, y_train)
    DTscores.append(DTclf.score(X_test, y_test))
    ############# for RF #################
```

```
    RFclf.fit(X_train, y_train)
    RFscores.append(RFclf.score(X_test, y_test))
    ############# For SVM #################
    SVclf.fit(X_train, y_train)
    SVscores.append(SVclf.score(X_test, y_test))

print('LRscores:', np.mean(LRscores))
print('DTscores:', np.mean(DTscores))
print('RFscores:', np.mean(RFscores))
print('SVscores:', np.mean(SVscores))
```

Result

K-Fold Cross Validation for LR, DT, RF & SVM:
LRscores: 0.91
DTscores: 0.89
RFscores: 0.91
SVscores: 0.98

Interpretation

The best technique among the four techniques using K-fold Corss-Validation (k=5) is Support Vector Machines as it has the highest score of 0.98.

3.iii Find the confusion matrix and ROC curve for the best method. Hence, calculate and interpret: Predictive value positive and negative, Accuracy, Sensitivity, and Specificity of the test.

Python Codes

#Confusion matrix for Support Vector Machines

```
from sklearn.svm import SVC
SVclf = SVC(kernel='poly', degree=4)
### kernel='linear', Gaussian kernel: kernel = 'rbf', kernel='sigmoid'

SVclf.fit(X_train, y_train)
y_pred=SVclf.predict(X_test)
print('Confusion Matrix:\n', confusion_matrix(y_test, y_pred))
print('Accuracy Score:', accuracy_score(y_test, y_pred))
```

#Calculating Predictive value positive and negative, Accuracy, Sensitivity, and Specificity of the test

```
# confusion matrix of SVM
cm = confusion_matrix(y_test, y_pred)

# calculating sensitivity from confusion matrix
sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )

# calculating specificity from confusion matrix
specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)

# calculating predictive value positive confusion matrix
predictive_value_posive = cm[0,0]/(cm[0,0]+cm[1,0])
print('Predictive value positive : ', predictive_value_posive)

# calculating predictive value negative from confusion matrix
predictive_value_negative = cm[1,1]/(cm[1,1]+cm[0,1])
print('Predictive value negative : ', predictive_value_negative)
```

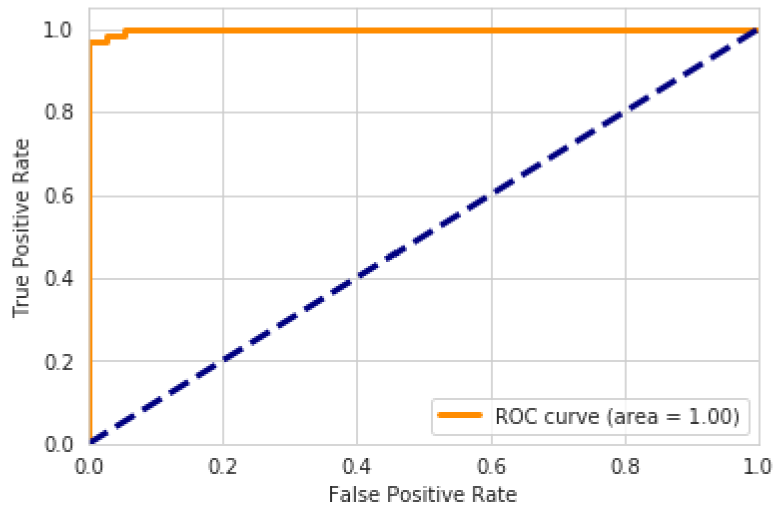
 # ROC CURVE

```
# Split the data into train and test sub-datasets
cols = ['Channel','Region','Fresh','Milk','Grocery','Frozen','Detergents_Paper', 'Delicassen']
X = df_new[cols]
y = df_new['Customer_Type']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.4, random_state=0)

# fit SVM model on the train data
from sklearn.svm import SVC
model = SVC(kernel='poly', degree=4)
model.fit(X_train, y_train)

# predict probabilities for the test data
probs = model.decision_function(X_test)

# compute the AUC Score
from sklearn.metrics import roc_curve, roc_auc_score
auc = roc_auc_score(y_test, probs)
print('AUC:', auc)

# get the ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, probs, pos_label='1')
```

```
# Plot ROC Curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=3, label='ROC curve (area = %0.2f)' % auc)
plt.plot([0, 1], [0, 1], color='navy', lw=3, linestyle='--')
# axis limits
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend(loc="lower right")
# plt.title('Receiver operating characteristic example')
# show the plot
plt.show()
```

Result

```
Confusion Matrix of SVM Model:
 [[48  1]
 [ 2 38]]

Accuracy Score of SVM: 0.97

Sensitivity of SVM:  0.98

Specificity of SVM :  0.95

Predictive value positive of SVM:  0.96

Predictive value negative of SVM:  0.97

AUC score of SVM: 0.9988
```



ROC Curve for SVM Model

Interpretation

For SVM Model:
Accuracy: 97% of predictions are correct.
Sensitivity: For all instances that were actually positive, 98% percent was classified correctly.
Specificity: For all instances that were actually negative, 95% percent was classified correctly.
Predictive value positive: For all instances classified positive, 96% was correct.
Predictive value negative: For all instances classified negative, 97% was correct.