Aadith Lasar

LY CORE-2

2203262

MACHINE LEARNING LAB-2

Title: Download the any dataset from UCI or Data.org or from any other data repositories and perform the basic data pre-processing steps using python/R .

After completion of this experiment students will be able to:

- - Learn to pre-process dataset

-Learn to use pandas and sklearn

Aim: To download the any dataset from UCI or Data.org or from any other data repositories and perform the basic data pre-processing steps using python/R .

Theory:

Step 1 : Import the libraries

Step 2 : Import the data-set

Step 3 : Check out the missing values

Step 4 : See the Categorical Values

Step 5 : Splitting the data-set into Training and Test Set

Data cleaning:

The main aim of Data Cleaning is to identify and remove errors & duplicate data, in order to create a reliable dataset. This improves the quality of the training data for analytics and enables accurate decision-making.

Needless to say, data cleansing is a time-consuming process and most data scientists spend an enormous amount of time in enhancing the quality of the data. However, there are various methods to identify and classify data for data cleansing

There are mainly two distinct techniques, namely Qualitative and Quantitative techniques to classify data errors. Qualitative techniques involve rules, constraints, and patterns to identify errors.

On the other hand, Quantitative techniques employ statistical techniques to identify errors in the trained data.

Normalisation:

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max scaling.

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Here, Xmax and Xmin are the maximum and the minimum values of the feature respectively.

- When the value of X is the minimum value in the column, the numerator will be 0, and hence X' is 0
- On the other hand, when the value of X is the maximum value in the column, the numerator is equal to the denominator and thus the value of X' is 1
- If the value of X is between the minimum and the maximum value, then the value of X' is between 0 and 1

Standardisation:

Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

Here's the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

μ is the mean of the feature values and σ is the standard deviation of the feature values. Note that in this case, the values are not restricted to a particular range.

X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.2,random_state= 0)

Split arrays or matrices into random train and test subsets Quick utility that wraps input validation and next(ShuffleSplit().split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

Imputer(missing_values='NaN', strategy='mean', axis=0)

Imputation transformer for completing missing values.

pandas.read_csv()

Read a comma-separated values (csv) file into DataFrame.

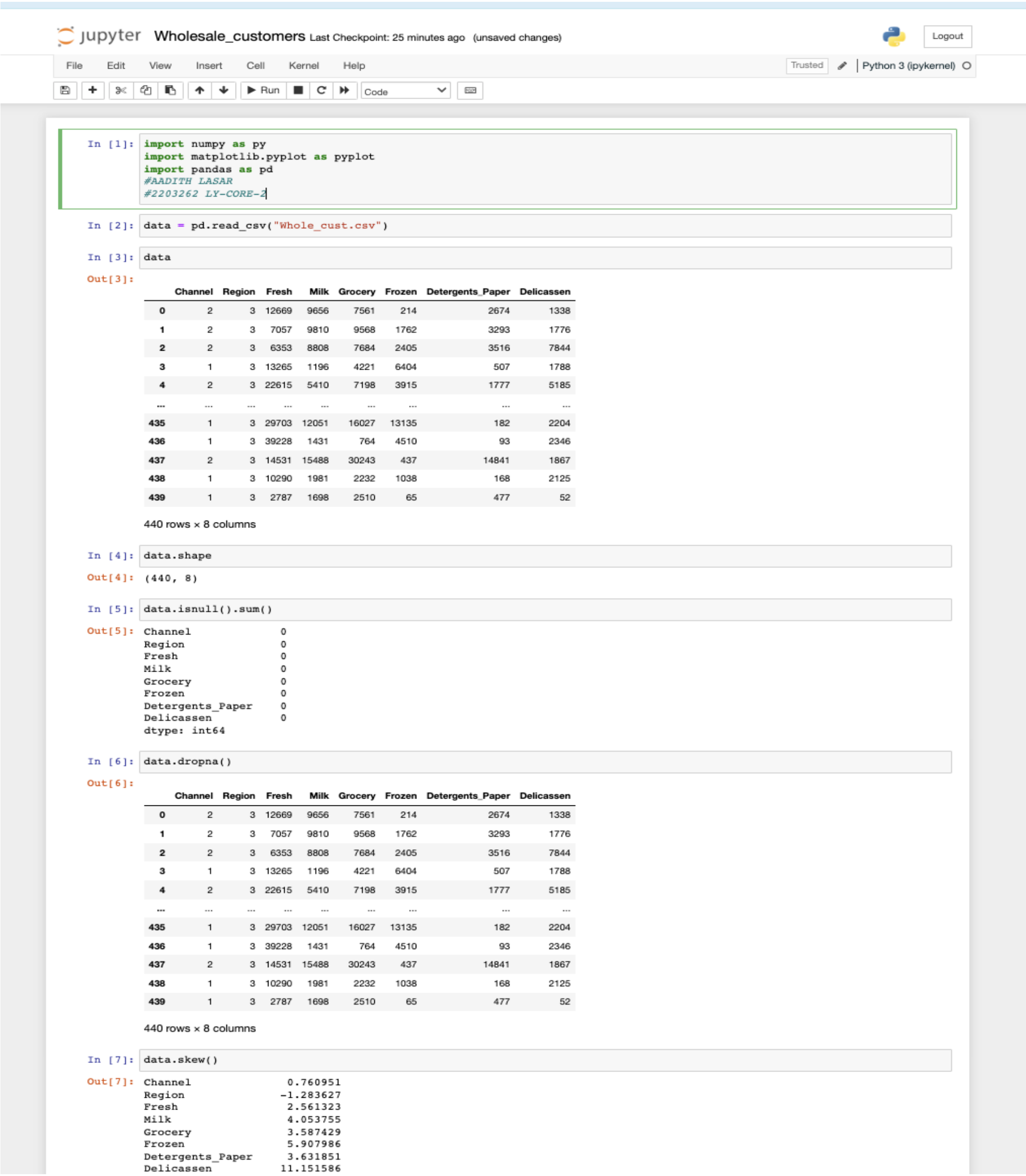
Also supports optionally iterating or breaking of the file into chunks.

Program:-

```
import numpy as py
import matplotlib.pyplot as pyplot
import pandas as pd

data = pd.read_csv("Whole_cust.csv")

data.shape

data.isnull().sum()

data.dropna()

data.skew()

data.kurtosis()

import seaborn as sns

sns.boxplot(x="Channel", y="Region", data=data)

data.plot(kind="scatter", x="Grocery", y="Detergents_Paper")
```

```
sns.violinplot(x="Channel", y="Region", data=data)\

sns.pairplot(data, hue="Fresh", height=4)

data.corr()

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

from sklearn.naive_bayes import GaussianNB
nvclassifier = GaussianNB()
nvclassifier.fit(X_train, y_train)

y_pred = nvclassifier.predict(X_test)
print(y_pred)

y_compare = py.vstack((y_test,y_pred)).T
y_compare[:5,:]

from sklearn.metrics import accuracy_score, confusion_matrix

acc = accuracy_score(y_test, y_pred) * 100
print("Accuracy =", acc)

confusion_matrix(y_test, y_pred)

from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred))
```

Output:-



```
In [1]: import numpy as py
        import matplotlib.pyplot as pyplot
        import pandas as pd
        #AADITH LASAR
        #2203262 LY-CORE-2
```

```
In [2]: data = pd.read_csv("Whole_cust.csv")
```

```
In [3]: data
```

Out[3]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185
...	...	...	...	...	...	...	...	...
435	1	3	29703	12051	16027	13135	182	2204
436	1	3	39228	1431	764	4510	93	2346
437	2	3	14531	15488	30243	437	14841	1867
438	1	3	10290	1981	2232	1038	168	2125
439	1	3	2787	1698	2510	65	477	52

440 rows × 8 columns

```
In [4]: data.shape

Out[4]: (440, 8)
```

```
In [5]: data.isnull().sum()

Out[5]: Channel             0
        Region              0
        Fresh               0
        Milk                0
        Grocery             0
        Frozen              0
        Detergents_Paper    0
        Delicassen          0
        dtype: int64
```

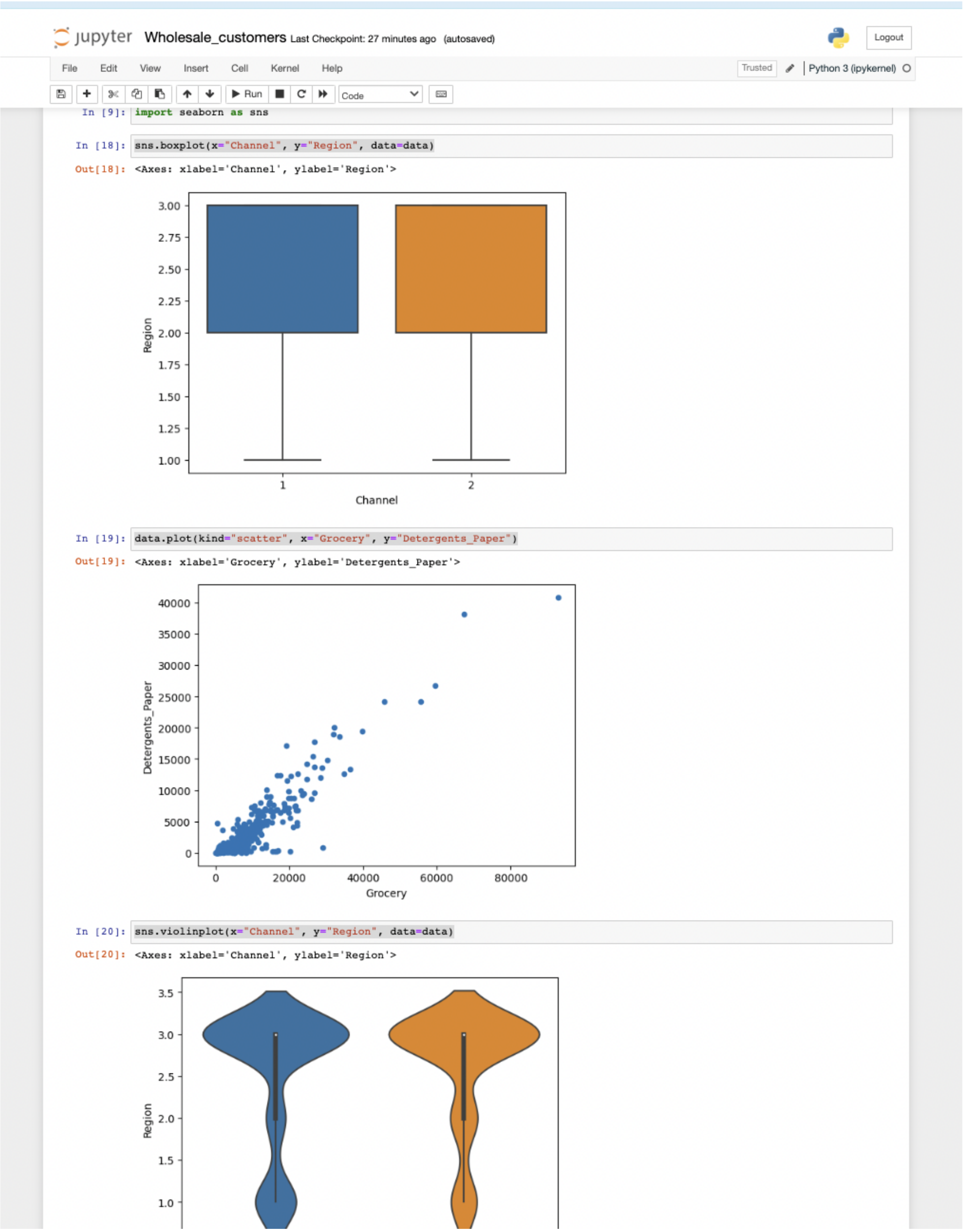
```
In [6]: data.dropna()
```

Out[6]:

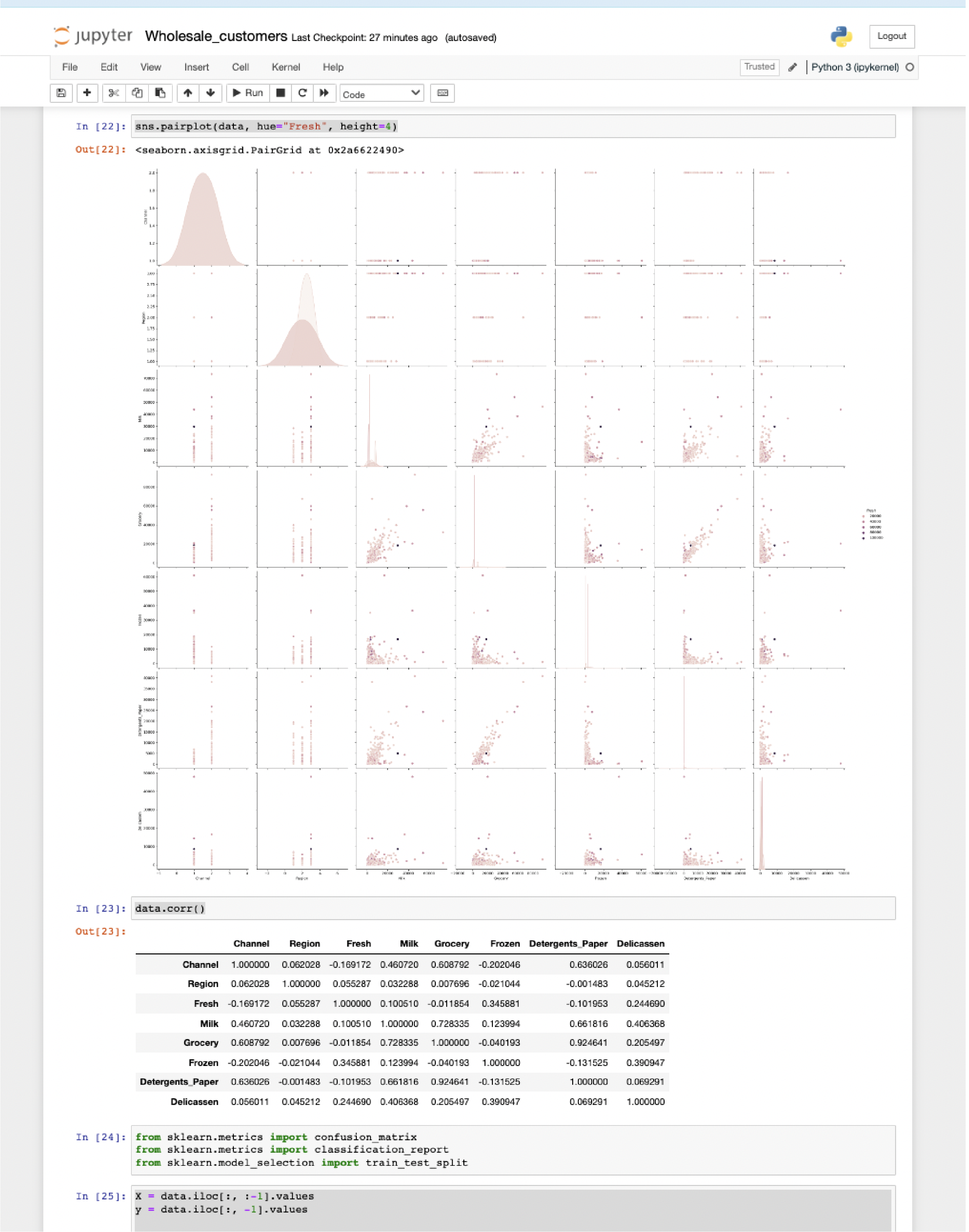
	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185
...	...	...	...	...	...	...	...	...
435	1	3	29703	12051	16027	13135	182	2204
436	1	3	39228	1431	764	4510	93	2346
437	2	3	14531	15488	30243	437	14841	1867
438	1	3	10290	1981	2232	1038	168	2125
439	1	3	2787	1698	2510	65	477	52

440 rows × 8 columns

```
In [7]: data.skew()

Out[7]: Channel              0.760951
        Region              -1.283627
        Fresh                2.561323
        Milk                 4.053755
        Grocery              3.587429
        Frozen               5.907986
        Detergents_Paper     3.631851
        Delicassen          11.151586
```



```
In [9]: import seaborn as sns

In [18]: sns.boxplot(x="Channel", y="Region", data=data)

Out[18]: <Axes: xlabel='Channel', ylabel='Region'>

In [19]: data.plot(kind="scatter", x="Grocery", y="Detergents_Paper")

Out[19]: <Axes: xlabel='Grocery', ylabel='Detergents_Paper'>

In [20]: sns.violinplot(x="Channel", y="Region", data=data)

Out[20]: <Axes: xlabel='Channel', ylabel='Region'>
```



In [22]:
```
sns.pairplot(data, hue="Fresh", height=4)
```

Out[22]:
```
<seaborn.axisgrid.PairGrid at 0x2a6622490>
```

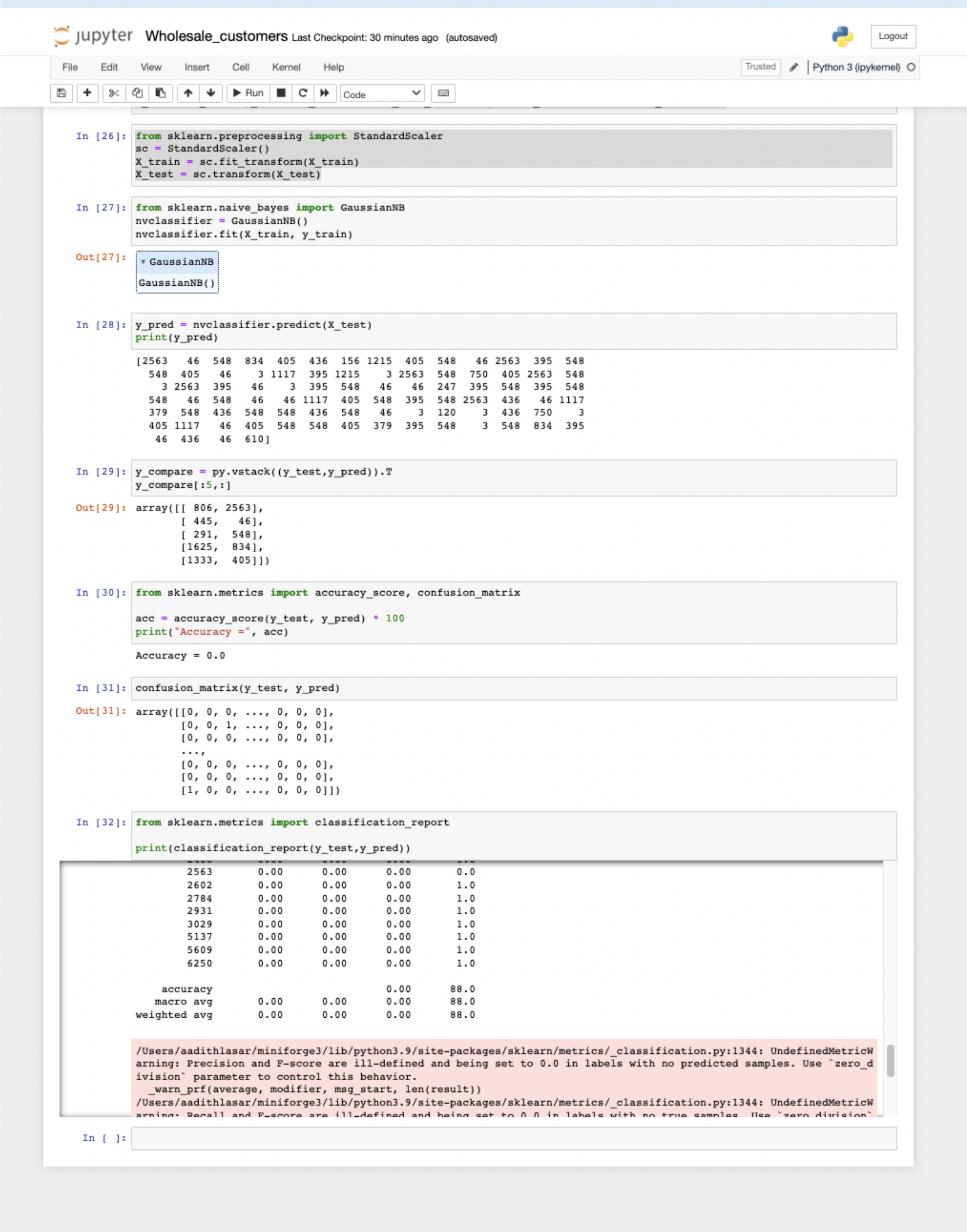
In [23]:
```
data.corr()
```

Out[23]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
Channel	1.000000	0.062028	-0.169172	0.460720	0.608792	-0.202046	0.636026	0.056011
Region	0.062028	1.000000	0.055287	0.032288	0.007696	-0.021044	-0.001483	0.045212
Fresh	-0.169172	0.055287	1.000000	0.100510	-0.011854	0.345881	-0.101953	0.244690
Milk	0.460720	0.032288	0.100510	1.000000	0.728335	0.123994	0.661816	0.406368
Grocery	0.608792	0.007696	-0.011854	0.728335	1.000000	-0.040193	0.924641	0.205497
Frozen	-0.202046	-0.021044	0.345881	0.123994	-0.040193	1.000000	-0.131525	0.390947
Detergents_Paper	0.636026	-0.001483	-0.101953	0.661816	0.924641	-0.131525	1.000000	0.069291
Delicassen	0.056011	0.045212	0.244690	0.406368	0.205497	0.390947	0.069291	1.000000

In [24]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

In [25]:
```
X = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
```



```
In [26]: from sklearn.preprocessing import StandardScaler
         sc = StandardScaler()
         X_train = sc.fit_transform(X_train)
         X_test = sc.transform(X_test)

In [27]: from sklearn.naive_bayes import GaussianNB
         nvclassifier = GaussianNB()
         nvclassifier.fit(X_train, y_train)

Out[27]: ▾ GaussianNB
         GaussianNB()

In [28]: y_pred = nvclassifier.predict(X_test)
         print(y_pred)

         [2563   46  548  834  405  436  156 1215  405  548   46 2563  395  548
           548  405   46    3 1117  395 1215    3 2563  548  750  405 2563  548
             3 2563  395   46    3  395  548   46   46  247  395  548  395  548
           548   46  548   46   46 1117  405  548  395  548 2563  436   46 1117
           379  548  436  548  548  436  548   46    3  120    3  436  750    3
           405 1117   46  405  548  548  405  379  395  548    3  548  834  395
            46  436   46  610]

In [29]: y_compare = py.vstack((y_test,y_pred)).T
         y_compare[:5,:]

Out[29]: array([[ 806, 2563],
                [ 445,   46],
                [ 291,  548],
                [1625,  834],
                [1333,  405]])

In [30]: from sklearn.metrics import accuracy_score, confusion_matrix

         acc = accuracy_score(y_test, y_pred) * 100
         print("Accuracy =", acc)

         Accuracy = 0.0

In [31]: confusion_matrix(y_test, y_pred)

Out[31]: array([[0, 0, 0, ..., 0, 0, 0],
                [0, 0, 1, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                ...,
                [0, 0, 0, ..., 0, 0, 0],
                [0, 0, 0, ..., 0, 0, 0],
                [1, 0, 0, ..., 0, 0, 0]])

In [32]: from sklearn.metrics import classification_report

         print(classification_report(y_test,y_pred))

                 2563       0.00      0.00      0.00       0.0
                 2602       0.00      0.00      0.00       1.0
                 2784       0.00      0.00      0.00       1.0
                 2931       0.00      0.00      0.00       1.0
                 3029       0.00      0.00      0.00       1.0
                 5137       0.00      0.00      0.00       1.0
                 5609       0.00      0.00      0.00       1.0
                 6250       0.00      0.00      0.00       1.0

             accuracy                           0.00      88.0
            macro avg       0.00      0.00      0.00      88.0
         weighted avg       0.00      0.00      0.00      88.0

         /Users/aadithlasar/miniforge3/lib/python3.9/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricW
         arning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_d
         ivision` parameter to control this behavior.
           _warn_prf(average, modifier, msg_start, len(result))
         /Users/aadithlasar/miniforge3/lib/python3.9/site-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricW
         arning: Recall and F-score are ill-defined and being set to 0.0 in labels with no true samples. Use `zero_division`

In [ ]:
```

Conclusion:

Thus we have successfully implemented pre-processing operations on a dataset