Kingdome of Saudi Arabia
Ministry of Higher Education
Taibah University
College of Computer Science and Engineering
Department of Information Systems

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة
كلية علوم وهندسة الحاسب الآلي
قسم نظم المعلومات

IS372 – Data warehouse and data mining

DATE: 24 May 2022

Project Title: Dresses Recommendation Analysis

Section: IA

Professor name: Mohammed alsarem

Team members-Id:

Areej Osama shareefi - 4050788

Aya Nasser Aloufi-4051793

Shroug sultan alharbi - 4050436

Content

Abstract

One of the most common problems that most girls face is finding the perfect dress on the internet and discovering their personal style, which is a problem that must be addressed. There are many girls who not only struggle to find the perfect dress online, but also refuse to go shopping because they do not want to wait in line or for a variety of other reasons.

As a result, we choose to research and solve this problem. One of the variables employed in solving this problem is inserting dress details such as the nick line, sleeve length, waistline, materials and fabric type, decoration, and pattern type as well as the season of the dress if it is winter, summer, spring, or autumn dress.

Introduction

Knowing the customer’s taste is a problem for most merchants, as bringing clothes that the customer does not like will cause their loss, so we will analyze the dresses and what is recommended to be brought and what is not recommended based on the price, rating, the season of the dress, and style of dresses. We will do this using the classification on the recommendation column, and we will build a decision tree that will be used in the future and will help us know whether bringing this type of dresses is a good option or not, which will solve the problem of losses and profits for the company.

The data on which we will conduct this analysis is data for sales of dresses for an online store, what is their style, price,season, rating, and whether they are recommended or not, which we obtained from https://archive.ics.uci.edu/ml/datasets/Dresses_Attribute_Sales.

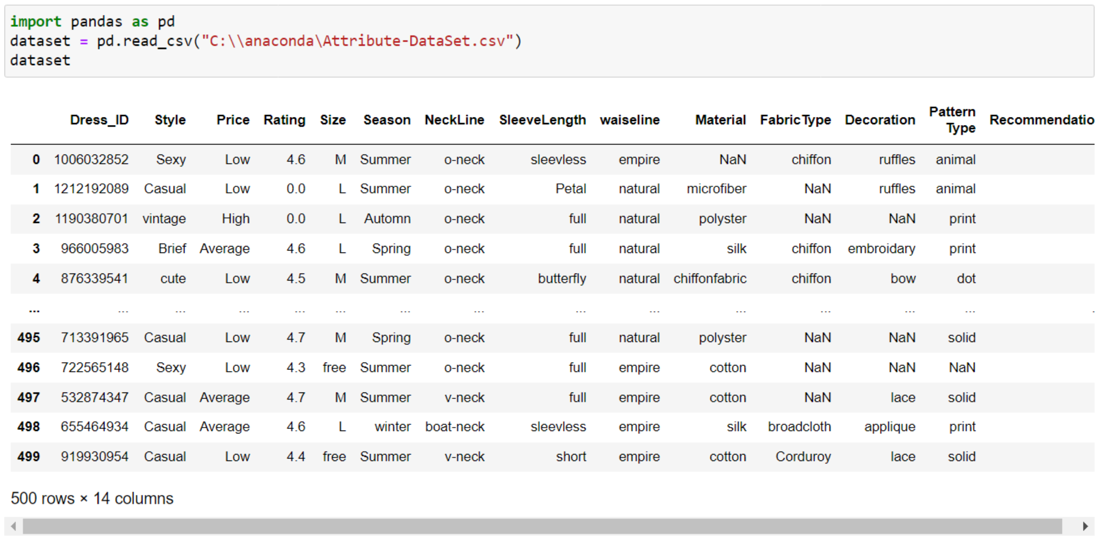
Methods

We used in this project Python to prospect in our dataset by anaconda notebook jupyter.

The name of our dataset is "Dresses_Attribute_Sales" In phase preprocessing, data cleaning and data processing we installed the libraries necessary [pandas as pd, numpy as np, sklearn]

and in graphics and analyzing phase we installed and import [graphviz ,from sklearn.preprocessing import LabelEncoder, from sklearn.tree import (DecisionTreeClassifier , export_graphviz), from sklearn.model_selection import train_test_split, from sklearn.metrics import (accuracy_score ,confusion_matrix ,classification_report ,mean_squared_error,mean_absolute_error) ,seaborn as sns ,matplotlib.pyplot as plt, From graphviz import Source, pydot: for resize tree figure, import os, from sklearn.linear_model import LinearRegression]

Dataset Description

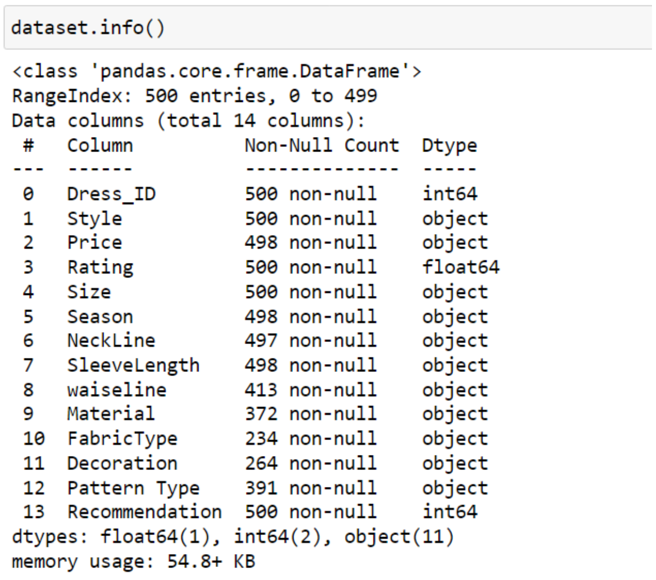

```
import pandas as pd
dataset = pd.read_csv("C:\\anaconda\Attribute-DataSet.csv")
dataset
```

	Dress_ID	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	Material	FabricType	Decoration	Pattern Type	Recommendatio
0	1006032852	Sexy	Low	4.6	M	Summer	o-neck	sleevless	empire	NaN	chiffon	ruffles	animal	
1	1212192089	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	microfiber	NaN	ruffles	animal	
2	1190380701	vintage	High	0.0	L	Automn	o-neck	full	natural	polyster	NaN	NaN	print	
3	966005983	Brief	Average	4.6	L	Spring	o-neck	full	natural	silk	chiffon	embroidary	print	
4	876339541	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chiffonfabric	chiffon	bow	dot	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
495	713391965	Casual	Low	4.7	M	Spring	o-neck	full	natural	polyster	NaN	NaN	solid	
496	722565148	Sexy	Low	4.3	free	Summer	o-neck	full	empire	cotton	NaN	NaN	NaN	
497	532874347	Casual	Average	4.7	M	Summer	v-neck	full	empire	cotton	NaN	lace	solid	
498	655464934	Casual	Average	4.6	L	winter	boat-neck	sleevless	empire	silk	broadcloth	applique	print	
499	919930954	Casual	Low	4.4	free	Summer	v-neck	short	empire	cotton	Corduroy	lace	solid	

500 rows × 14 columns

First of all, we started with importing the dataset so we can do our research and analysis on it.

As we can see the dataset has 500 rows and 14 column and the dataset is about the dresses in the website and the details of dress like the nick line and sleeve length waiseline and the materials, fabric type, decoration pattern type and the season of the dress if it is winter dress or summer or spring or autumn.


```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Dress_ID        500 non-null    int64
 1   Style           500 non-null    object
 2   Price           498 non-null    object
 3   Rating          500 non-null    float64
 4   Size            500 non-null    object
 5   Season          498 non-null    object
 6   NeckLine        497 non-null    object
 7   SleeveLength    498 non-null    object
 8   waiseline       413 non-null    object
 9   Material        372 non-null    object
 10  FabricType      234 non-null    object
 11  Decoration      264 non-null    object
 12  Pattern Type    391 non-null    object
 13  Recommendation  500 non-null    int64
dtypes: float64(1), int64(2), object(11)
memory usage: 54.8+ KB
```

Second thing we want to know more information about our dataset, so we used info() method to print more information. The information contains the number of columns, column labels, column data types and the number of cells in each column.

```
dataset.isnull().any()
```

```
Dress_ID          False
Style             False
Price              True
Rating            False
Size              False
Season             True
NeckLine           True
SleeveLength       True
waiseline          True
Material           True
FabricType         True
Decoration         True
Pattern Type       True
Recommendation    False
dtype: bool
```

As we can see some of the data has null values, so we need to do data cleaning to get accurate result by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.

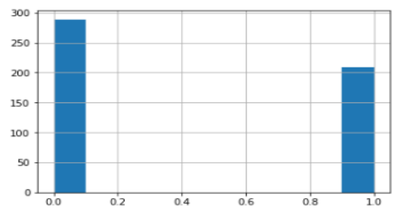
```
dataset.isnull().sum()
```

```
Dress_ID            0
Style               0
Price               2
Rating              0
Size                0
Season              2
NeckLine            3
SleeveLength        2
waiseline          87
Material          128
FabricType        266
Decoration        236
Pattern Type      109
Recommendation      0
dtype: int64
```

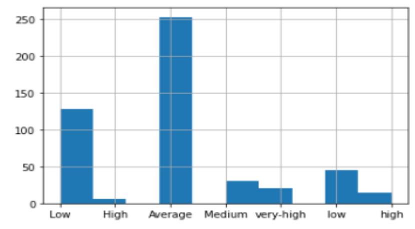
The column price has 2 null value ,season has 2 null value , neckline has 3 null value, sleevelength has 2 null value ,waiseline,materials,fabrictype,decoration,patterntype all these columns have null values we need to work on cleaning these columns to get the accurate result.

```
import matplotlib.pyplot as plt
dataset['Recommendation'].hist()
```

```
<AxesSubplot:>
```

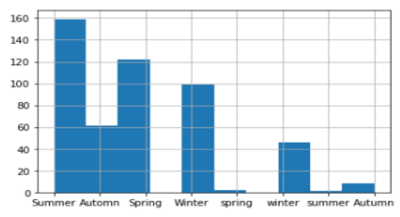


```
<AxesSubplot:>
```



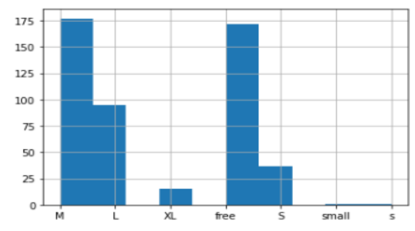
```
dataset['Season'].hist()
```

```
<AxesSubplot:>
```



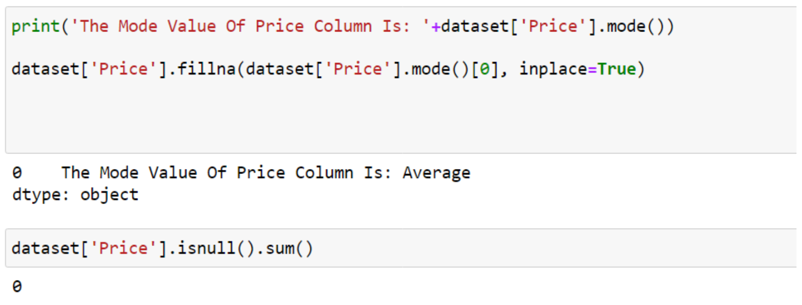
```
dataset['Size'].hist()
```

```
<AxesSubplot:>
```



We observed that some records have the same meaning but different name as the season summer is the same Summer, and the size S is the same of small So we will standardize the names in order to get an accurate result

Data cleaning



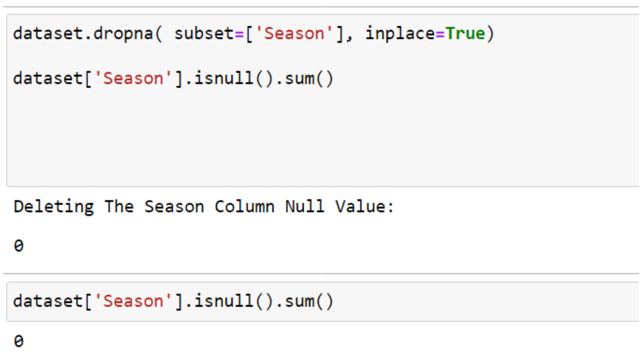
```
print('The Mode Value Of Price Column Is: '+dataset['Price'].mode())

dataset['Price'].fillna(dataset['Price'].mode()[0], inplace=True)
```

```
0    The Mode Value Of Price Column Is: Average
dtype: object
```

```
dataset['Price'].isnull().sum()
```

```
0
```

Now to get accurate analysis we should first deal with the missing values so , as we see in the previous figure the price has two null values so we fill it with the mode of the column price which is the 'Average' by using fillna() and mode() method and we check the null value in price column to make sure the code worked fine and it did.



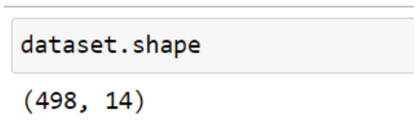
```
dataset.dropna( subset=['Season'], inplace=True)

dataset['Season'].isnull().sum()
```

```
Deleting The Season Column Null Value:

0
```

```
dataset['Season'].isnull().sum()
```

```
0
```

We delete the rows which has null value for Season column.



```
dataset.shape
```

```
(498, 14)
```

After deleting the rows which has null value in season column we now have 498 row and 14 column, and we knew by using shape method which indicate the number of rows and column in the dataframe .

```
print('The Mode Value Of NeckLine Column Is: '+dataset['NeckLine'].mode())
dataset['NeckLine'].fillna(dataset['NeckLine'].mode()[0], inplace=True)

print('The Mode Value Of SleeveLength Column Is: '+dataset['SleeveLength'].mode())
dataset['SleeveLength'].fillna(dataset['SleeveLength'].mode()[0], inplace=True)

print('The Mode Value Of waiseline Column Is: '+dataset['waiseline'].mode())
dataset['waiseline'].fillna(dataset['waiseline'].mode()[0], inplace=True)

print('The Mode Value Of Material Column Is: '+dataset['Material'].mode())
dataset['Material'].fillna(dataset['Material'].mode()[0], inplace=True)

print('The Mode Value Of FabricType Column Is: '+dataset['FabricType'].mode())
dataset['FabricType'].fillna(dataset['FabricType'].mode()[0], inplace=True)

print('The Mode Value Of Decoration Column Is: '+dataset['Decoration'].mode())
dataset['Decoration'].fillna(dataset['Decoration'].mode()[0], inplace=True)

print('The Mode Value Of Pattern Type  Column Is: '+dataset["Pattern Type"].mode())
dataset["Pattern Type"].fillna(dataset["Pattern Type"].mode()[0], inplace=True)
```

```
0    The Mode Value Of NeckLine Column Is: o-neck
dtype: object
0    The Mode Value Of SleeveLength Column Is: slee...
dtype: object
0    The Mode Value Of waiseline Column Is: natural
dtype: object
0    The Mode Value Of Material Column Is: cotton
dtype: object
0    The Mode Value Of FabricType Column Is: chiffon
dtype: object
0    The Mode Value Of Decoration Column Is: lace
dtype: object
0    The Mode Value Of Pattern Type  Column Is: solid
dtype: object
```

We fill the column which have a null values by the mode.

```
dataset.isnull().sum()
```

```
Dress_ID          0
Style             0
Price             0
Rating            0
Size              0
Season            0
NeckLine          0
SleeveLength      0
waiseline         0
Material          0
FabricType        0
Decoration        0
Pattern Type      0
Recommendation    0
dtype: int64
```

Now we used the isnull().sum() method that count the null values to know if there is any column with null values and as we can see there isn't.

	Style	Price	Rating	Size	Season	NeckLine	SleeveLength	waiseline	Material	FabricType	Decoration	Pattern Type	Recommendation
0	Sexy	Low	4.6	M	Summer	o-neck	sleevless	empire	cotton	chiffon	ruffles	animal	1
1	Casual	Low	0.0	L	Summer	o-neck	Petal	natural	microfiber	chiffon	ruffles	animal	0
2	vintage	High	0.0	L	Automn	o-neck	full	natural	polyster	chiffon	lace	print	0
3	Brief	Average	4.6	L	Spring	o-neck	full	natural	silk	chiffon	embroidary	print	1
4	cute	Low	4.5	M	Summer	o-neck	butterfly	natural	chiffonfabric	chiffon	bow	dot	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
495	Casual	Low	4.7	M	Spring	o-neck	full	natural	polyster	chiffon	lace	solid	1
496	Sexy	Low	4.3	free	Summer	o-neck	full	empire	cotton	chiffon	lace	solid	0
497	Casual	Average	4.7	M	Summer	v-neck	full	empire	cotton	chiffon	lace	solid	1
498	Casual	Average	4.6	L	winter	boat-neck	sleevless	empire	silk	broadcloth	applique	print	1
499	Casual	Low	4.4	free	Summer	v-neck	short	empire	cotton	Corduroy	lace	solid	0

498 rows × 13 columns

As long as we don't need the DressID column in our analysis we drop it so, we have now 498 rows and 13 column.

```
dataset.describe()
```

```
           Dress_ID       Rating  Recommendation
count  4.980000e+02  498.000000      498.000000
mean   9.058026e+08    3.533133        0.419679
std    1.738268e+08    2.002334        0.494003
min    4.442820e+08    0.000000        0.000000
25%    7.681081e+08    3.775000        0.000000
50%    9.083296e+08    4.600000        0.000000
75%    1.039833e+09    4.800000        1.000000
max    1.253973e+09    5.000000        1.000000
```

```
import numpy as np
dataset.describe(include=[np.object])
```

```
C:\Users\dell\AppData\Local\Temp/ipykernel_21092/3380137737.py:2: DeprecationWarning: `np.object` is a deprecated alias for
the builtin `object`. To silence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  dataset.describe(include=[np.object])
```

```
         Style    Price  Size  Season  NeckLine  SleeveLength  waiseline  Material  FabricType  Decoration  Pattern Type
count      498      498   498     498       498           498        498       498         498         498           498
unique      13        7     7       8        16            17          4        23          22          23            14
top     Casual  Average     M  Summer    o-neck     sleevless    natural    cotton     chiffon        lace         solid
freq       231      253   177     159       273           224        390       278         399         305           311
```

For descriptive or summary statistics we used describe() method. That gives the mean, std and IQR values.

Generally, describe() function excludes the character columns and gives summary statistics of numeric columns

So we needed to add a variable named include='objective' to get the summary the descriptive statistics for character column.

```
dataset['Season'].replace("winter",
            "Winter",
            inplace=True)

dataset['Season'].replace("summer",
            "Summer",
            inplace=True)

dataset['Season'].replace("Automn",
            "Autumn",
            inplace=True)

dataset['Season'].replace("spring",
            "Spring",
            inplace=True)

dataset['Price'].replace("low",
            "Low",
            inplace=True)

dataset['Price'].replace("high",
            "High",
            inplace=True)

dataset['Size'].replace("s",
            "S",
            inplace=True)

dataset['Size'].replace("small",
            "S",
            inplace=True)
```

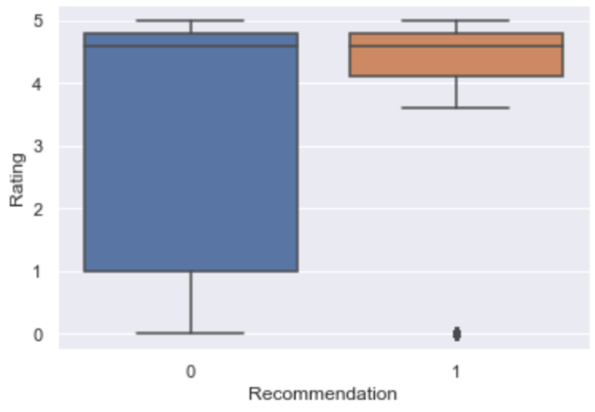
We standardized the names so the result will be more accurate by using replace method that replace value with value you choose.

Data visualization

Box plot and Violin Plots

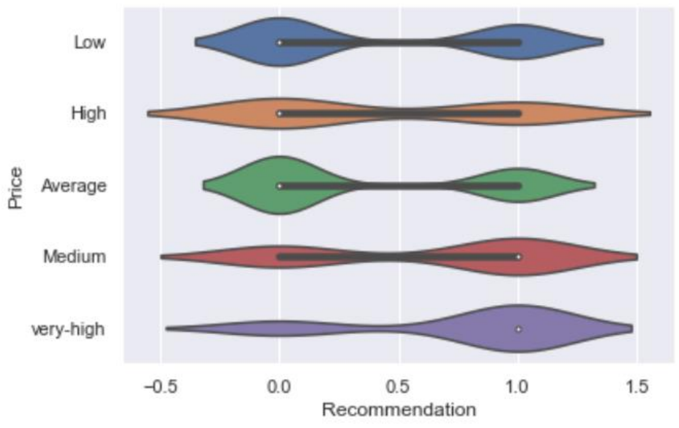
We used Boxplot and Violin Plots to show the relationships between a category's rating (recommendation) and other attributes it might have to do with it.

```
In [308]:  sns.boxplot(x='Recommendation',y='Rating',data=dataset)
           plt.show

Out[308]: <function matplotlib.pyplot.show(close=None, block=None)>
```

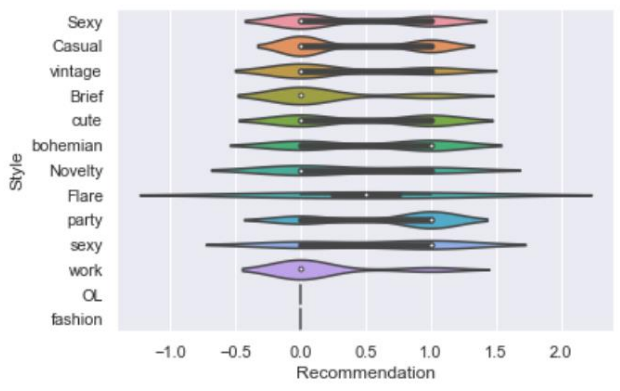


In the Boxplot, we establish a relationship with Rating and showed that the values of the unrecommended products range from 0-5, and the values of the recommended products range from 3.7 to 5.

```
In [309]:  sns.violinplot(x='Recommendation',y='Price',data=dataset)
           plt.show

Out[309]: <function matplotlib.pyplot.show(close=None, block=None)>
```



```
In [310]:  sns.violinplot(x='Recommendation',y='Style',data=dataset)
           plt.show

Out[310]: <function matplotlib.pyplot.show(close=None, block=None)>
```

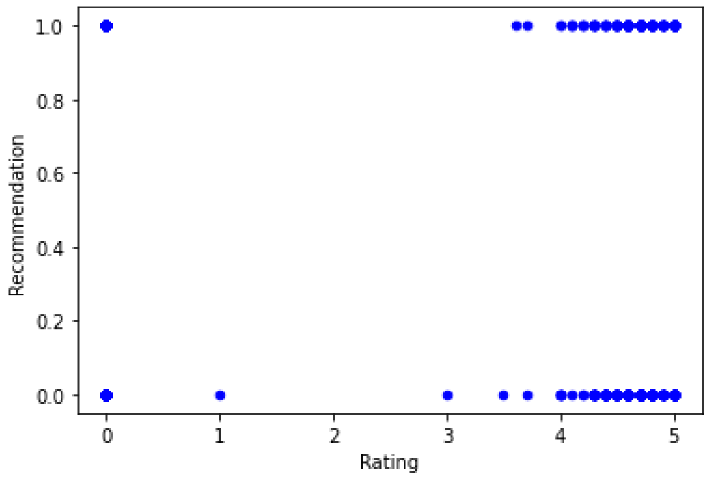


In the second and third figure we used Violin Plots because it shows our data better and clearer. It appears from the second figure that the relationship was with the pric e and the third figure was with the style and the plot of the figure increases at the va lues 1 and 0 in an axis X with the increase in the number of values in an axis Y

Scatter plots:

A scatter plot is a form of plot or mathematical diagram that displays values for two variables for a collection of data using Cartesian coordinates. Dots are used to indicate values for two different variables in a scatter plot. The values for each data point are indicated by the position of each dot on the horizontal and vertical axes. Relationships between variables are visualized using scatter plots.

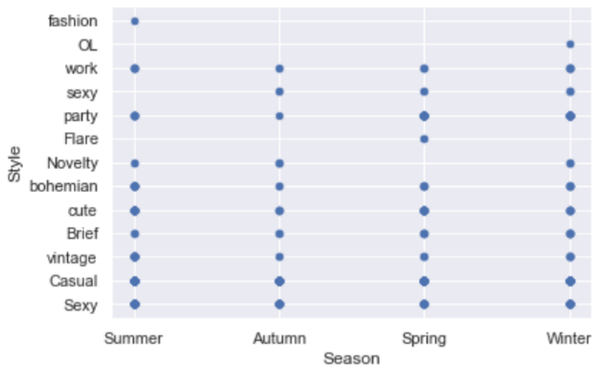
```
import matplotlib.pyplot as plt
dataset.plot(kind='scatter',x='Rating',y='Recommendation',c='b')
plt.show()
```



In the scatter plot above we can see the relationship between column Rating and column Recommendation. As shown we can see that there is a higher rating(4-5) in the recommendation 0 and 1.

```
dataset.plot(kind="scatter" ,x="Season" , y="Style" , c='b')
```

```
<AxesSubplot:xlabel='Season', ylabel='Style'>
```



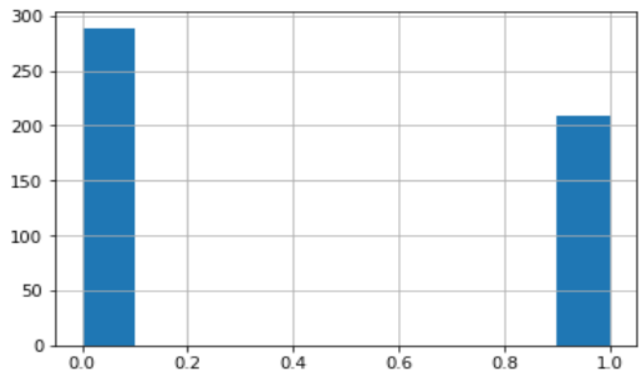
In this scatter plots above we can see the values for the two variable season and the style which shows us the styles that we have in each season.

Histogram plots:

A histogram is a graph showing *frequency* distributions of the values.

```
import matplotlib.pyplot as plt
dataset['Recommendation'].hist()
```

```
<AxesSubplot:>
```

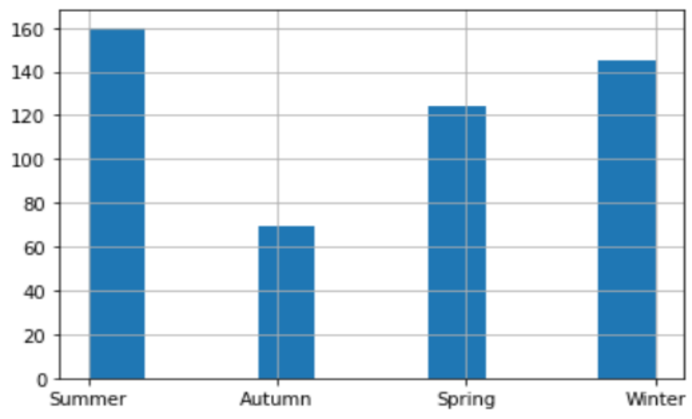


In this figure we observe that the recommendation which is if the dress is recommended or not to buy has two value 0 for not recommended and 1 for the recommended dresses

So , we notice that 0 has almost 280 and 1 has almost 210 which the mean the recommended dresses is less that the dresses that is not recommended.

```
dataset['Season'].hist()
```

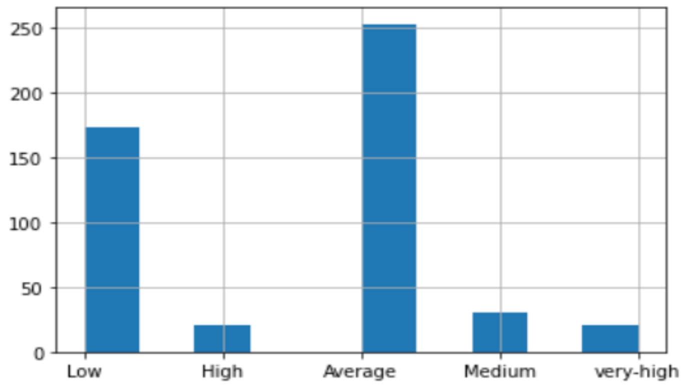
```
<AxesSubplot:>
```



We observed that the summer season has the most dresses in the website with 160 dress and after that come winter with almost 140 dress ,and then the spring with almost 120 and the last is the autumn with 70 dress.

```
dataset['Price'].hist()
```

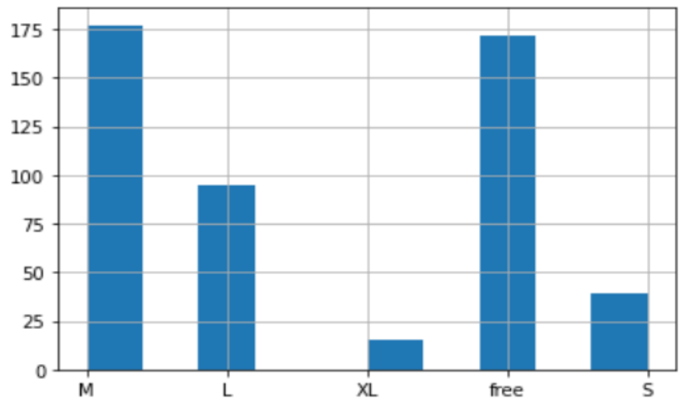
```
<AxesSubplot:>
```



There is 5 value of the column price (low,medium,high,very high,average) and the most price we got is the average with almost 250 dress and the less price is the high and very high which they close in result.

```
dataset['Size'].hist()
```

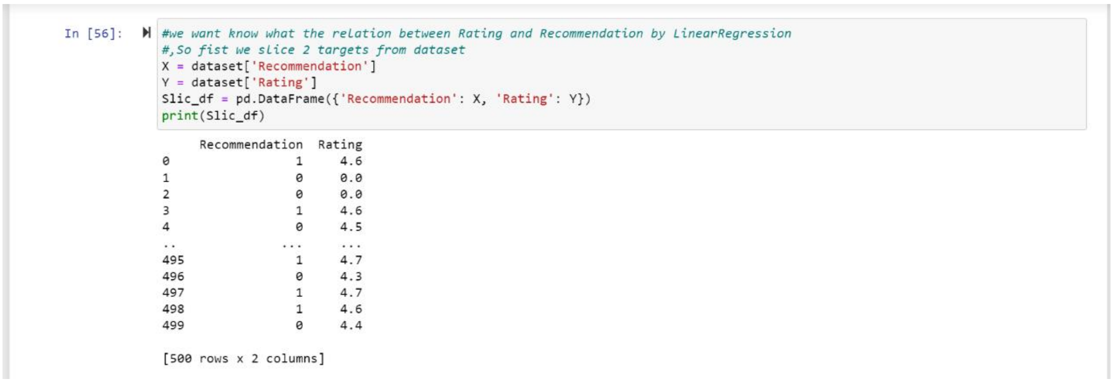
```
<AxesSubplot:>
```



We see that there 5 result of Size column which is(S,M,L,Xl,Freesize) and the most size we got is the medium and the less size the XL.

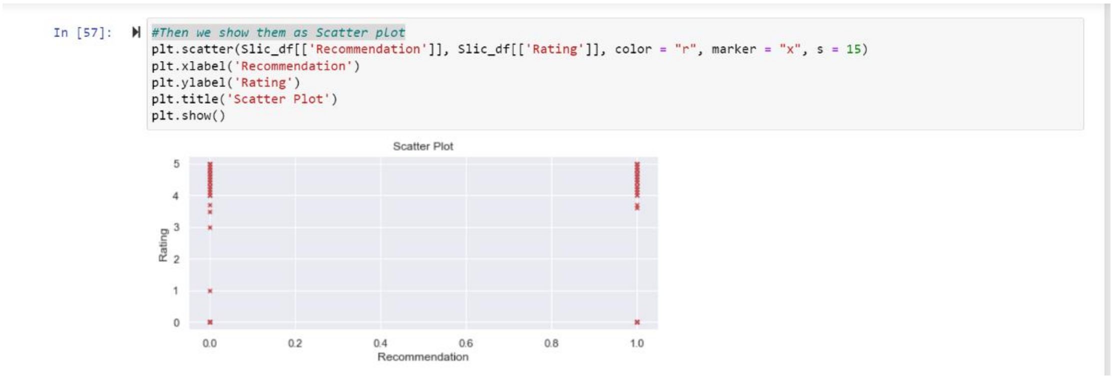
Linear Regression

The "Recommendation" is the target class in our project, so we tried to know what the most influential category is, and from this point, we will tried the "Rating" as a most influential category and we showed that by applied Linear Regression to it.



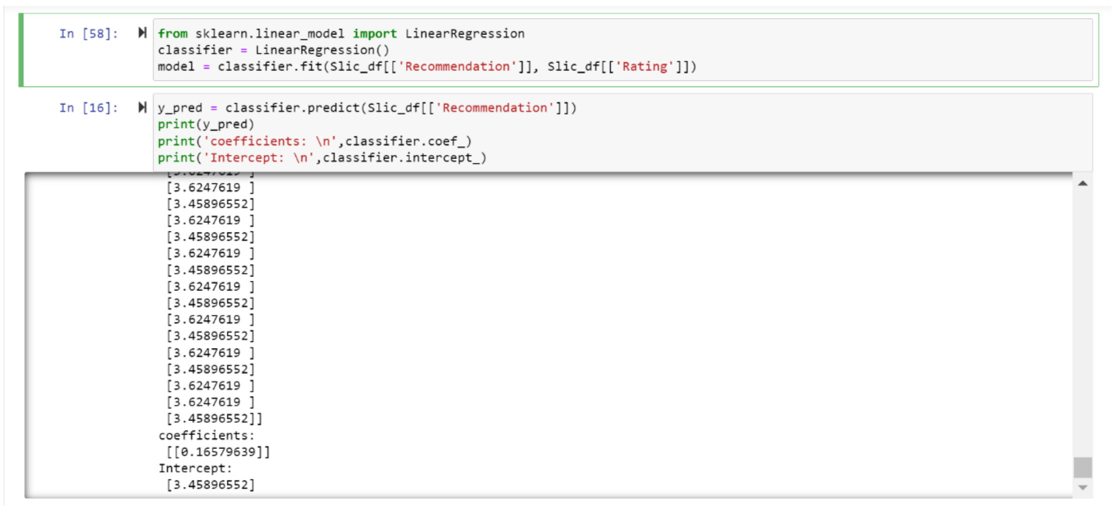
```
In [56]:  #we want know what the relation between Rating and Recommendation by LinearRegression
          #,So fist we slice 2 targets from dataset
          X = dataset['Recommendation']
          Y = dataset['Rating']
          Slic_df = pd.DataFrame({'Recommendation': X, 'Rating': Y})
          print(Slic_df)

           Recommendation  Rating
      0                 1     4.6
      1                 0     0.0
      2                 0     0.0
      3                 1     4.6
      4                 0     4.5
      ..              ...     ...
      495               1     4.7
      496               0     4.3
      497               1     4.7
      498               1     4.6
      499               0     4.4

      [500 rows x 2 columns]
```

So, fist we sliced the 2 targets (Recommendation, Rating) from dataset.



```
In [57]:  #Then we show them as Scatter plot
          plt.scatter(Slic_df[['Recommendation']], Slic_df[['Rating']], color = "r", marker = "x", s = 15)
          plt.xlabel('Recommendation')
          plt.ylabel('Rating')
          plt.title('Scatter Plot')
          plt.show()
```

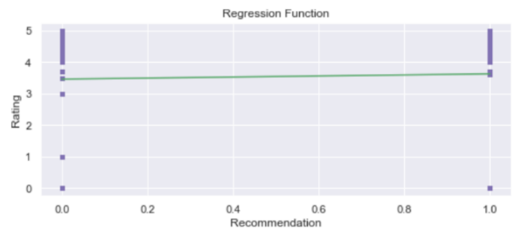
Then we showed them as Scatter plot.



```
In [58]:  from sklearn.linear_model import LinearRegression
          classifier = LinearRegression()
          model = classifier.fit(Slic_df[['Recommendation']], Slic_df[['Rating']])

In [16]:  y_pred = classifier.predict(Slic_df[['Recommendation']])
          print(y_pred)
          print('coefficients: \n',classifier.coef_)
          print('Intercept: \n',classifier.intercept_)

 [3.6247619 ]
 [3.45896552]
 [3.6247619 ]
 [3.45896552]
 [3.6247619 ]
 [3.45896552]
 [3.6247619 ]
 [3.45896552]
 [3.6247619 ]
 [3.45896552]
 [3.6247619 ]
 [3.45896552]
 [3.6247619 ]
 [3.6247619 ]
 [3.45896552]]
coefficients:
 [[0.16579639]]
Intercept:
 [3.45896552]
```

We built Linear Regression and calculated the coefficients: 0.16579639 and Intercept: 3.45896552 for the target class (Recommendation)

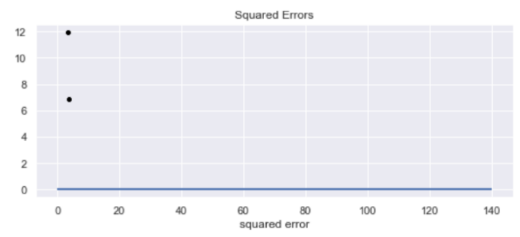
```
In [59]:  plt.scatter(Slic_df[['Recommendation']], Slic_df[['Rating']], color = "m", marker = "s", s = 10)
          plt.plot(Slic_df[['Recommendation']],y_pred,color='g')
          plt.xlabel('Recommendation')
          plt.ylabel('Rating')
          plt.title('Regression Function')
          plt.show()
```



after that we visualized our regression function with the scatterplot showed the original data set.

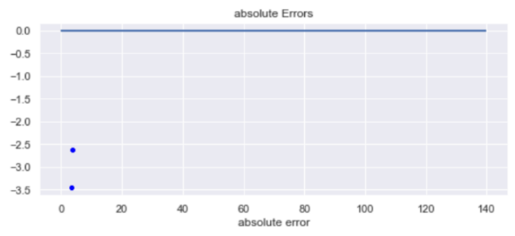
In [62]:
```python
from sklearn.metrics import mean_squared_error,mean_absolute_error

print('Mean squared error:')
print(mean_squared_error(Slic_df['Recommendation'],y_pred))

plt.scatter(y_pred,(Slic_df[['Recommendation']] - y_pred) ** 2,color="black",s=10)
plt.title("Squared Errors")

plt.hlines(y=0,xmin=0,xmax=140,linewidth=2)
plt.xlabel('predicted values')
plt.xlabel('squared error')
plt.show()
```

```
Mean squared error:
9.83291414449918
```



In [61]:
```python
print('Mean absolute error:')
print(mean_absolute_error(Slic_df['Recommendation'],y_pred))

plt.scatter(y_pred,(Slic_df[['Recommendation']] - y_pred) ,color="blue",s=10)
plt.title("absolute Errors")

plt.hlines(y=0,xmin=0,xmax=140,linewidth=2)
plt.xlabel('predicted values')
plt.xlabel('absolute error')
plt.show()
```

```
Mean absolute error:
3.1085999999999996
```



To evaluating the regression model, we computed their error (Mean squared error, Mean absolute error)

Mean squared error: 9.83291414449918

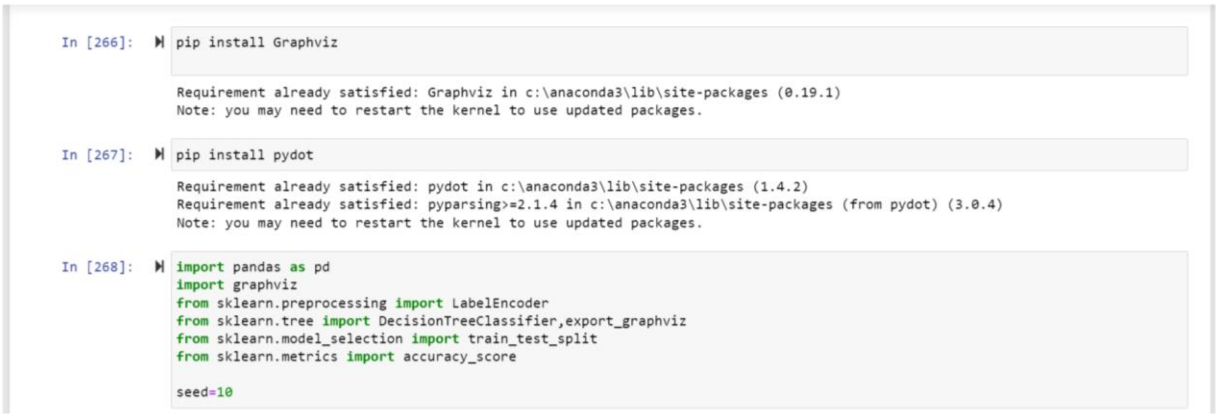
Mean absolute error: 3.1085999999999996

Both errors gave small values, so the error in the model we built is almost non-existent.

Research Results

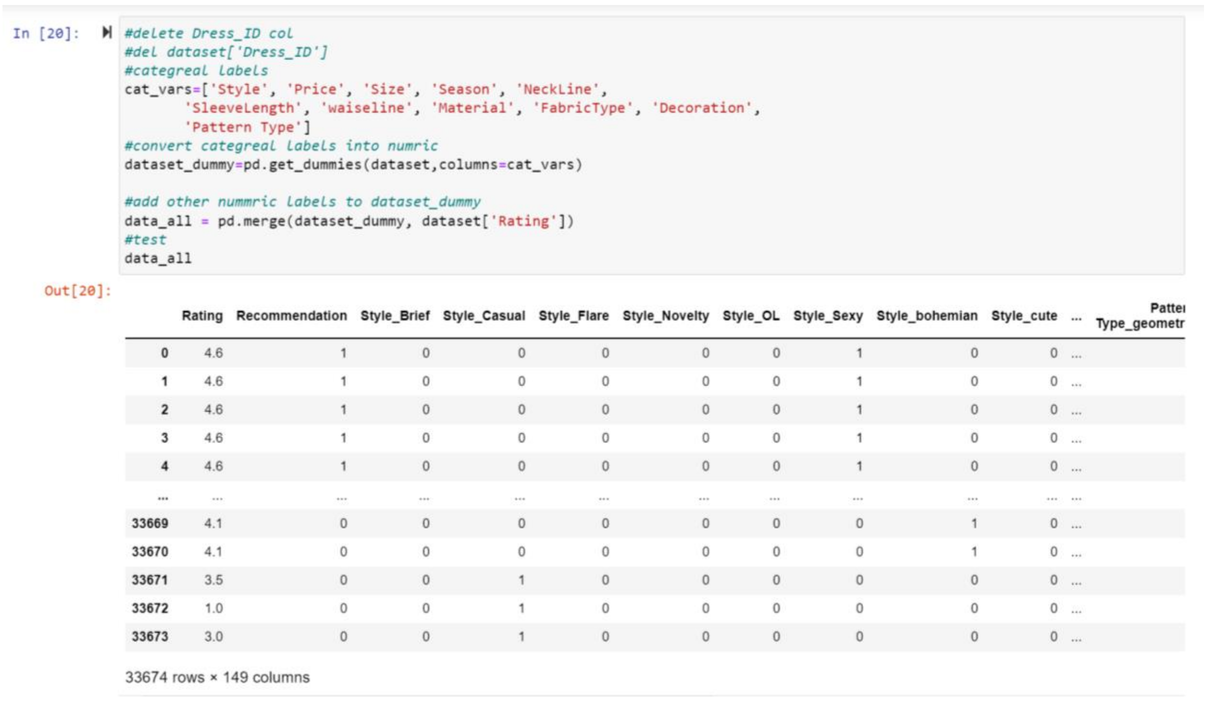
Decision Tree Classification

Using the Decision Tree Classification on our data will show the value of the Class Label (Recommendation) and this will help similar stores to identify recommended products from unrecommended ones.



```
In [266]:  pip install Graphviz

           Requirement already satisfied: Graphviz in c:\anaconda3\lib\site-packages (0.19.1)
           Note: you may need to restart the kernel to use updated packages.

In [267]:  pip install pydot

           Requirement already satisfied: pydot in c:\anaconda3\lib\site-packages (1.4.2)
           Requirement already satisfied: pyparsing>=2.1.4 in c:\anaconda3\lib\site-packages (from pydot) (3.0.4)
           Note: you may need to restart the kernel to use updated packages.

In [268]:  import pandas as pd
           import graphviz
           from sklearn.preprocessing import LabelEncoder
           from sklearn.tree import DecisionTreeClassifier,export_graphviz
           from sklearn.model_selection import train_test_split
           from sklearn.metrics import accuracy_score

           seed=10
```

We downloaded the graphviz library to be able to shape and display the tree, and we also downloaded the pydot library to control the size of the tree graph.[1]



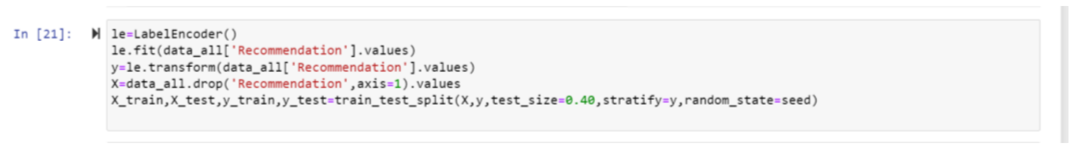
```
In [20]:  #delete Dress_ID col
          #del dataset['Dress_ID']
          #categreal labels
          cat_vars=['Style', 'Price', 'Size', 'Season', 'NeckLine',
                 'SleeveLength', 'waiseline', 'Material', 'FabricType', 'Decoration',
                 'Pattern Type']
          #convert categreal labels into numric
          dataset_dummy=pd.get_dummies(dataset,columns=cat_vars)

          #add other nummric labels to dataset_dummy
          data_all = pd.merge(dataset_dummy, dataset['Rating'])
          #test
          data_all
```

Out[20]:

	Rating	Recommendation	Style_Brief	Style_Casual	Style_Flare	Style_Novelty	Style_OL	Style_Sexy	Style_bohemian	Style_cute	...	Patte Type_geometr
0	4.6	1	0	0	0	0	0	1	0	0	...	
1	4.6	1	0	0	0	0	0	1	0	0	...	
2	4.6	1	0	0	0	0	0	1	0	0	...	
3	4.6	1	0	0	0	0	0	1	0	0	...	
4	4.6	1	0	0	0	0	0	1	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
33669	4.1	0	0	0	0	0	0	0	1	0	...	
33670	4.1	0	0	0	0	0	0	0	1	0	...	
33671	3.5	0	0	1	0	0	0	0	0	0	...	
33672	1.0	0	0	1	0	0	0	0	0	0	...	
33673	3.0	0	0	1	0	0	0	0	0	0	...	

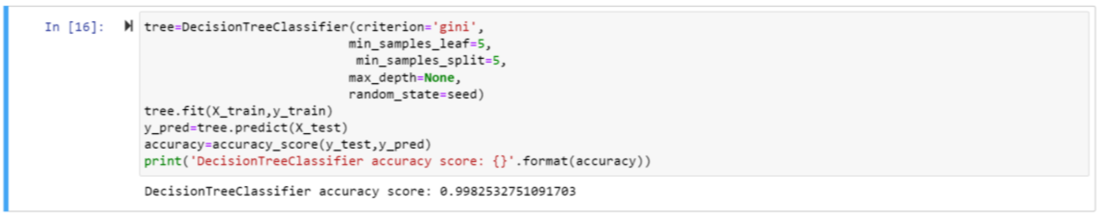
33674 rows × 149 columns

our dataset has categorical attributes, so we need to convert them to numeric values to calculate them in the first we wrote all categorical attributes in array then we use get_dummies () method [2] from pandas' library as a converter and made new dataset with converted categorical attributes after that we add numeric columns to new dataset(data_all).



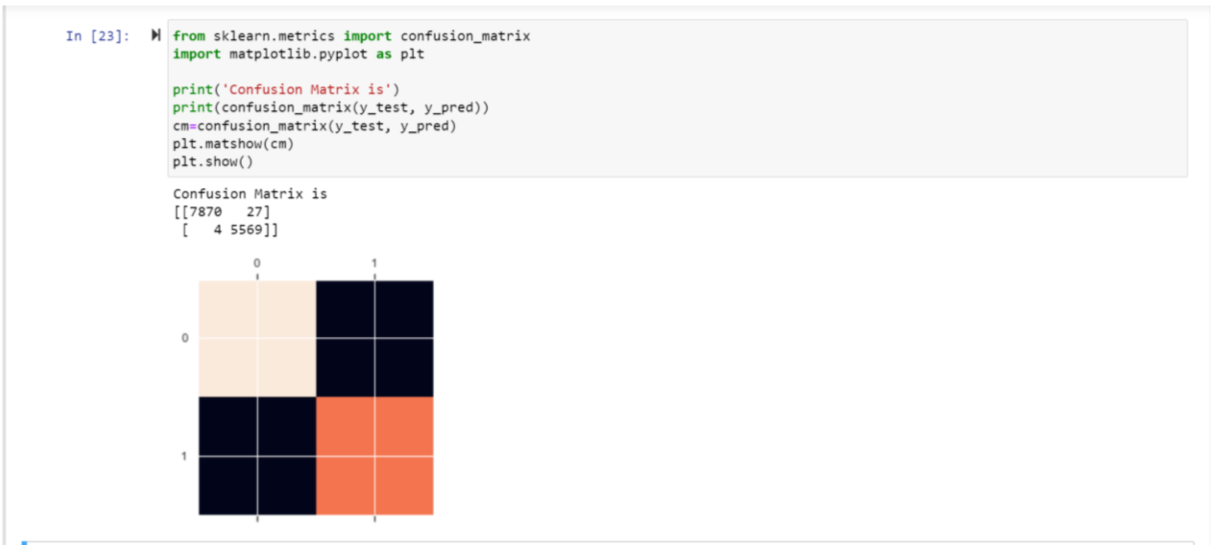
```
In [21]:  le=LabelEncoder()
          le.fit(data_all['Recommendation'].values)
          y=le.transform(data_all['Recommendation'].values)
          X=data_all.drop('Recommendation',axis=1).values
          X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.40,stratify=y,random_state=seed)
```

A part of the dataset that we used in this project was tested at 40%.

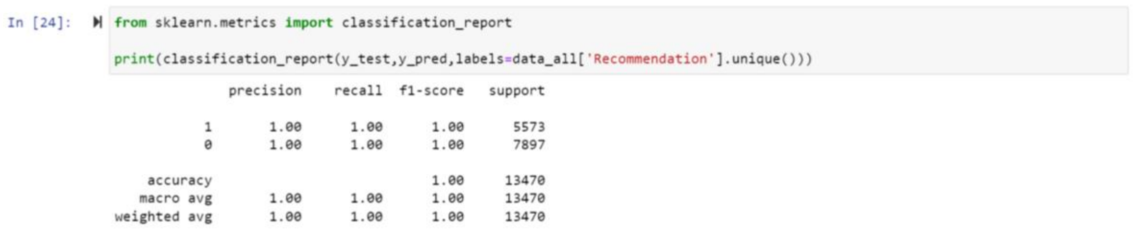


```
In [16]:  tree=DecisionTreeClassifier(criterion='gini',
                                       min_samples_leaf=5,
                                        min_samples_split=5,
                                       max_depth=None,
                                       random_state=seed)
          tree.fit(X_train,y_train)
          y_pred=tree.predict(X_test)
          accuracy=accuracy_score(y_test,y_pred)
          print('DecisionTreeClassifier accuracy score: {}'.format(accuracy))

          DecisionTreeClassifier accuracy score: 0.9982532751091703
```

The "entropy" equation was used to reach the accuracy equal to 99%.



```
In [23]:  from sklearn.metrics import confusion_matrix
          import matplotlib.pyplot as plt

          print('Confusion Matrix is')
          print(confusion_matrix(y_test, y_pred))
          cm=confusion_matrix(y_test, y_pred)
          plt.matshow(cm)
          plt.show()

          Confusion Matrix is
          [[7870   27]
           [   4 5569]]
```



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

          print(classification_report(y_test,y_pred,labels=data_all['Recommendation'].unique()))

                        precision    recall  f1-score   support

                     1       1.00      1.00      1.00      5573
                     0       1.00      1.00      1.00      7897

              accuracy                           1.00     13470
             macro avg       1.00      1.00      1.00     13470
          weighted avg       1.00      1.00      1.00     13470
```

and the confusion matrix was built to calculate the accuracy, precision, recall, f1-score, and support.

Now we calculate accuracy, precision, recall and F-measure

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy = (7870 + 5569) / (7870 + 5569+ 27 + 4) =13439/13470= 0.9976 ≈ 99%

$$\textbf{\textit{Precision}} = \frac{TP}{TP + FP}$$

Precision = (7870) / (7870 + 4) =7870/7874= 0.9994 ≈ 99%

$$\textbf{\textit{Recall}} = \frac{TP}{TP + FN}$$

Recall = (7870) / (7870 + 27) =7870/7897= 0.9965 ≈ 99%

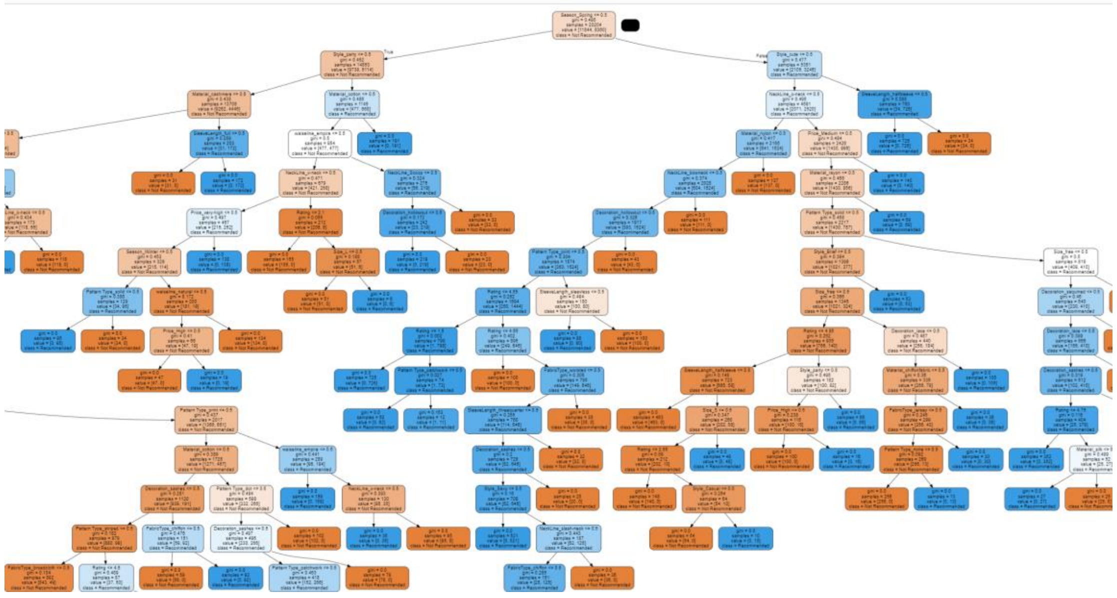
$$\textbf{\textit{F - measure}} = \frac{2*Recall*Precision}{Recall + Precision}$$

F1-score = (2*0.99 * 0.99) / (0.99 + 0.99) = 0.99 ≈ 99%

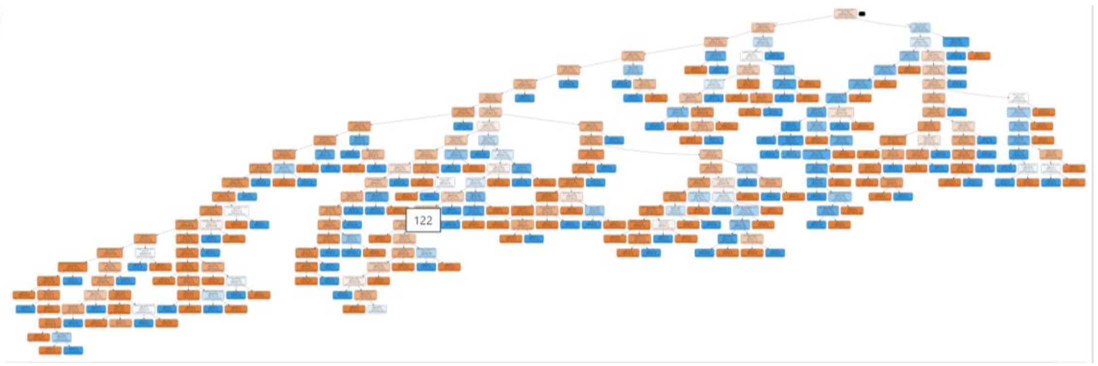
Decision Tree Classifier

In [32]:

```
from graphviz import Source
import pydot
import os

#installGraphviz3
os.environ["PATH"] += os.pathsep+ 'C:/Program Files/Graphviz/bin/'
def plot_tree(tree, dataframe, label_col, label_encoder, plot_title):
    label_names= ['Not Recommended', 'Recommended']
    graph_data= export_graphviz(tree,
                                feature_names=dataframe.drop(label_col, axis=1).columns,
                                class_names=label_names,
                                filled=True,
                                rounded=True,
                                out_file=None)
    #Generating plot and resizing tree graph.
    pdot = pydot.graph_from_dot_data(graph_data)
    # Access element [0] because graph_from_dot_data actually returns a list of DOT elements.
    pdot[0].set_graph_defaults(size = "\"10,10\"")
    graph = Source(pdot[0].to_string())
    graph.render(plot_title)
    return graph
tree_graph= plot_tree(tree, data_all, 'Recommendation', le, 'Attribute-DataSet')
tree_graph
```



]:



Conclusion

Finally, we believe we have developed an advanced classifier that will continue to provide details about the dress using artificial intelligence approaches such as deep searching, allowing us to apply it to real-world challenges this will help similar stores to identify recommended products from unrecommended ones.

To view the full project code and view the dataset, click the link:

https://github.com/AreejShareefi/project-Dresses.git

References

[1]"Matplotlib Histograms", *W3schools.com*, 2022. [Online]. Available: https://www.w3schools.com/python/matplotlib_histograms.asp. [Accessed: 24- May- 2022].

[2]"Descriptive or Summary Statistics in python pandas - describe() - DataScience Made Simple", *DataScience Made Simple*, 2022. [Online]. Available: https://www.datasciencemadesimple.com/descriptive-summary-statistics-python-pandas/. [Accessed: 24- May- 2022].

[3] https://stackoverflow.com/questions/49766875/trying-to-view-decision-tree-in-my-notebook

[4] https://towardsdatascience.com/understanding-decision-tree-classification-with-scikit-learn-2ddf272731bd