In [1]:
```
import openpyxl as xl
```

In [2]:
```
wb = xl.load_workbook("transactions.xlsx")
sheet = wb["Sheet1"]
cell = sheet["a1"]
cell = sheet.cell(1,1)              #cell method of sheet object to select the
cell a1
print(cell.value)
```

```
transaction_id
```

In [3]:
```
#display number of total row
print(sheet.max_row)
```

```
4
```

In [4]:
```
#getting total row sequence
for row in range(1, sheet.max_row + 1):
    print (row)
```

```
1
2
3
4
```

In [5]:
```
#pulling the column values
for row in range(2, sheet.max_row + 1):          #range started from row 2 to
exclude headers
    cell = sheet.cell(row, 3)                    #using sheet object to read
 values from column 3
    print (cell.value)
```

```
5.95
6.95
7.95
```

In [6]:
```
#adding a column with 10% correction(drop)
for row in range(2, sheet.max_row + 1):
    cell = sheet.cell(row, 3)
    corrected_price = cell.value * 0.9
    corrected_price_cell = sheet.cell(row, 4)                              #using
sheet object to call in the cell value from the column
    corrected_price_cell.value = corrected_price
#setting values for the cell
wb.save("transactions_updated.xlsx")
```

In [7]:
```
#inserting chart for the updated column values
from openpyxl.chart import BarChart, Reference

wb = xl.load_workbook("transactions.xlsx")
sheet = wb["Sheet1"]
for row in range(2, sheet.max_row + 1):
    cell = sheet.cell(row, 3)
    corrected_price = cell.value * 0.9
    corrected_price_cell = sheet.cell(row, 4)
    corrected_price_cell.value = corrected_price
                                                        #reference class u
sed to select the range of values
values = Reference(sheet,                               #inserting chart i
n the file
            min_row=2,
            max_row=sheet.max_row,
            min_col=4,
            max_col=4)

chart = BarChart()
chart.add_data(values)
sheet.add_chart(chart)

wb.save("transactions2.xlsx")
```

In [8]:
```
#Creating function to process multiple files
from openpyxl.chart import BarChart, Reference
def process_workbook(filename):
    wb = xl.load_workbook("filename")
    sheet = wb["Sheet1"]
    for row in range(2, sheet.max_row + 1):
        cell = sheet.cell(row, 3)
        corrected_price = cell.value * 0.9
        corrected_price_cell = sheet.cell(row, 4)
        corrected_price_cell.value = corrected_price
                                                            #reference cla
ss used to select the range of values
    values = Reference(sheet,                               #inserting cha
rt in the file
                min_row=2,
                max_row=sheet.max_row,
                min_col=4,
                max_col=4)

    chart = BarChart()
    chart.add_data(values)
    sheet.add_chart(chart)

    wb.save("filename")
```

In [9]:
```
#Machine Learning - Subset of AI
import pandas as pd
df = pd.read_csv('vgsales.csv')
df
```

Out[9]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26
6	7	New Super Mario Bros.	DS	2006.0	Platform	Nintendo	11.38	9.23
7	8	Wii Play	Wii	2006.0	Misc	Nintendo	14.03	9.20
8	9	New Super Mario Bros. Wii	Wii	2009.0	Platform	Nintendo	14.59	7.06
9	10	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63
10	11	Nintendogs	DS	2005.0	Simulation	Nintendo	9.07	11.00
11	12	Mario Kart DS	DS	2005.0	Racing	Nintendo	9.81	7.57
12	13	Pokemon Gold/Pokemon Silver	GB	1999.0	Role-Playing	Nintendo	9.00	6.18
13	14	Wii Fit	Wii	2007.0	Sports	Nintendo	8.94	8.03
14	15	Wii Fit Plus	Wii	2009.0	Sports	Nintendo	9.09	8.59
15	16	Kinect Adventures!	X360	2010.0	Misc	Microsoft Game Studios	14.97	4.94
16	17	Grand Theft Auto V	PS3	2013.0	Action	Take-Two Interactive	7.01	9.27
17	18	Grand Theft Auto: San Andreas	PS2	2004.0	Action	Take-Two Interactive	9.43	0.40
18	19	Super Mario World	SNES	1990.0	Platform	Nintendo	12.78	3.75
19	20	Brain Age: Train Your Brain in Minutes a Day	DS	2005.0	Misc	Nintendo	4.75	9.26
20	21	Pokemon Diamond/Pokemon Pearl	DS	2006.0	Role-Playing	Nintendo	6.42	4.52
21	22	Super Mario Land	GB	1989.0	Platform	Nintendo	10.83	2.71
22	23	Super Mario Bros. 3	NES	1988.0	Platform	Nintendo	9.54	3.44
23	24	Grand Theft Auto V	X360	2013.0	Action	Take-Two Interactive	9.63	5.31
24	25	Grand Theft Auto: Vice City	PS2	2002.0	Action	Take-Two Interactive	8.41	5.49

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales
25	26	Pokemon Ruby/Pokemon Sapphire	GBA	2002.0	Role-Playing	Nintendo	6.06	3.90
26	27	Pokemon Black/Pokemon White	DS	2010.0	Role-Playing	Nintendo	5.57	3.28
27	28	Brain Age 2: More Training in Minutes a Day	DS	2005.0	Puzzle	Nintendo	3.44	5.36
28	29	Gran Turismo 3: A-Spec	PS2	2001.0	Racing	Sony Computer Entertainment	6.85	5.09
29	30	Call of Duty: Modern Warfare 3	X360	2011.0	Shooter	Activision	9.03	4.28
...	...	...	...	...	...	...	...	...
16568	16571	XI Coliseum	PSP	2006.0	Puzzle	Sony Computer Entertainment	0.00	0.00
16569	16572	Resident Evil 4 HD	XOne	2016.0	Shooter	Capcom	0.01	0.00
16570	16573	Farming 2017 - The Simulation	PS4	2016.0	Simulation	UIG Entertainment	0.00	0.01
16571	16574	Grisaia no Kajitsu: La Fruit de la Grisaia	PSP	2013.0	Adventure	Prototype	0.00	0.00
16572	16575	Scarlett: Nichijou no Kyoukaisen	PS2	2008.0	Adventure	Kadokawa Shoten	0.00	0.00
16573	16576	Mini Desktop Racing	Wii	2007.0	Racing	Popcorn Arcade	0.01	0.00
16574	16577	Yattaman Wii: BikkuriDokkiri Machine de Mou Ra...	Wii	2008.0	Racing	Takara Tomy	0.00	0.00
16575	16578	Neo Angelique Special	PSP	2008.0	Adventure	Tecmo Koei	0.00	0.00
16576	16579	Rugby Challenge 3	XOne	2016.0	Sports	Alternative Software	0.00	0.01
16577	16580	Damnation	PC	2009.0	Shooter	Codemasters	0.00	0.01
16578	16581	Outdoors Unleashed: Africa 3D	3DS	2011.0	Sports	Mastiff	0.01	0.00
16579	16582	PGA European Tour	N64	2000.0	Sports	Infogrames	0.01	0.00
16580	16583	Real Rode	PS2	2008.0	Adventure	Kadokawa Shoten	0.00	0.00
16581	16584	Fit & Fun	Wii	2011.0	Sports	Unknown	0.00	0.01
16582	16585	Planet Monsters	GBA	2001.0	Action	Titus	0.01	0.00
16583	16586	Carmageddon 64	N64	1999.0	Action	Virgin Interactive	0.01	0.00

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales
16584	16587	Bust-A-Move 3000	GC	2003.0	Puzzle	Ubisoft	0.01	0.00
16585	16588	Breach	PC	2011.0	Shooter	Destineer	0.01	0.00
16586	16589	Secret Files 2: Puritas Cordis	DS	2009.0	Adventure	Deep Silver	0.00	0.01
16587	16590	Mezase!! Tsuri Master DS	DS	2009.0	Sports	Hudson Soft	0.00	0.00
16588	16591	Mega Brain Boost	DS	2008.0	Puzzle	Majesco Entertainment	0.01	0.00
16589	16592	Chou Ezaru wa Akai Hana: Koi wa Tsuki ni Shiru...	PSV	2016.0	Action	dramatic create	0.00	0.00
16590	16593	Eiyuu Densetsu: Sora no Kiseki Material Collec...	PSP	2007.0	Role-Playing	Falcom Corporation	0.00	0.00
16591	16594	Myst IV: Revelation	PC	2004.0	Adventure	Ubisoft	0.01	0.00
16592	16595	Plushees	DS	2008.0	Simulation	Destineer	0.01	0.00
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00

16598 rows × 11 columns

In [10]:
```
import pandas as pd
df = pd.read_csv('vgsales.csv')
df.shape
```

Out[10]: (16598, 11)

In [11]:
```
#Statistical review of data
df.describe()
```

Out[11]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Glo
count	16598.000000	16327.000000	16598.000000	16598.000000	16598.000000	16598.000000	165
mean	8300.605254	2006.406443	0.264667	0.146652	0.077782	0.048063	
std	4791.853933	5.828981	0.816683	0.505351	0.309291	0.188588	
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	
25%	4151.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	
50%	8300.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	
75%	12449.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	

In [13]:
```
df.values
```

Out[13]:
```
array([[1, 'Wii Sports', 'Wii', ..., 3.77, 8.46, 82.74],
       [2, 'Super Mario Bros.', 'NES', ..., 6.81, 0.77, 40.24],
       [3, 'Mario Kart Wii', 'Wii', ..., 3.79, 3.31, 35.82],
       ...,
       [16598, 'SCORE International Baja 1000: The Official Game', 'PS2',
        ..., 0.0, 0.0, 0.01],
       [16599, 'Know How 2', 'DS', ..., 0.0, 0.0, 0.01],
       [16600, 'Spirits & Spells', 'GBA', ..., 0.0, 0.0, 0.01]],
      dtype=object)
```

In [14]:
```
#Loading data for project music prediction for users
music_data = pd.read_csv("music.csv")
music_data
```

Out[14]:

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

In [15]:
```
#Preparing the data - splitting data into input set and output set
music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
X
```

Out[15]:

	age	gender
0	20	1
1	23	1
2	25	1
3	26	1
4	29	1
5	30	1
6	31	1
7	33	1
8	37	1
9	20	0
10	21	0
11	25	0
12	26	0
13	27	0
14	30	0
15	31	0
16	34	0
17	35	0

In [16]:
```
music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
y = music_data['genre']
y
```

Out[16]:
```
0         HipHop
1         HipHop
2         HipHop
3           Jazz
4           Jazz
5           Jazz
6      Classical
7      Classical
8      Classical
9          Dance
10         Dance
11         Dance
12      Acoustic
13      Acoustic
14      Acoustic
15     Classical
16     Classical
17     Classical
Name: genre, dtype: object
```

In [17]:
```
#Buidling model
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
y = music_data['genre']

#create object model
model = DecisionTreeClassifier()
#model includes two datasets inpute and output
model.fit(X, y)
#inspecting data
music_data
```

Out[17]:

	age	gender	genre
0	20	1	HipHop
1	23	1	HipHop
2	25	1	HipHop
3	26	1	Jazz
4	29	1	Jazz
5	30	1	Jazz
6	31	1	Classical
7	33	1	Classical
8	37	1	Classical
9	20	0	Dance
10	21	0	Dance
11	25	0	Dance
12	26	0	Acoustic
13	27	0	Acoustic
14	30	0	Acoustic
15	31	0	Classical
16	34	0	Classical
17	35	0	Classical

In [18]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
y = music_data['genre']

#create object model
model = DecisionTreeClassifier()
#model includes two datasets inpute and output
model.fit(X, y)
#based on assumption making prediction in variable
predictions = model.predict([ [21, 1], [22, 0]])
predictions
```

Out[18]:
```
array(['HipHop', 'Dance'], dtype=object)
```

In [19]:
```
#calculating the accuracy
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
y = music_data['genre']

#assignming 20% data for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

#create object model
model = DecisionTreeClassifier()
#passing training dataset
model.fit(X_train, y_train)
#instead of passing sample passing test dataset
predictions = model.predict(X_test)

#calling accuracy function giving two arguments for expected value and actual
 values will return accuracy score between 0-1
score = accuracy_score(y_test, predictions)
score
```

Out[19]:
```
1.0
```

In [20]:
```
#Persisting Models - creating a reusable trained model
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
#joblib object has methods for saving and loading modules
from sklearn.externals import joblib

#splitting dataset
music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
y = music_data['genre']

#training model
model = DecisionTreeClassifier()
#passing training dataset
model.fit(X, y)

#calling joblib with two arguments model and name of file to store this model
joblib.dump(model, "music-recommender.joblib")

#just storing the train model in a file
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\joblib\__init__.
py:15: DeprecationWarning: sklearn.externals.joblib is deprecated in 0.21 and
will be removed in 0.23. Please import this functionality directly from jobli
b, which can be installed with: pip install joblib. If this warning is raised
when loading pickled models, you may need to re-serialize those models with s
cikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

Out[20]:
```
['music-recommender.joblib']
```

In [21]:
```
#Persisting Models - loading a reusable trained model
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
#joblib object has methods for saving and loading modules
from sklearn.externals import joblib

# #splitting dataset
# music_data = pd.read_csv("music.csv")
# X = music_data.drop(columns=["genre"])
# y = music_data['genre']

# #training model
# model = DecisionTreeClassifier()
# #passing training dataset
# model.fit(X, y)

#Calling Load method using model name this returs train model
model = joblib.load("music-recommender.joblib")

#Making prediction
predictions = model.predict([[21, 1]])
predictions
```

Out[21]:
```
array(['HipHop'], dtype=object)
```

In [22]:
```
#Visualizing decision tree - exporting model in visual format for prediction
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
#Tree object has methods of exporting our decision tree into graphical format
from sklearn import tree

#splitting dataset to create input and output dataset
music_data = pd.read_csv("music.csv")
X = music_data.drop(columns=["genre"])
y = music_data['genre']

#create and train model
model = DecisionTreeClassifier()
#passing training dataset
model.fit(X, y)

#after model is trained called in tree to export graphical view using argument
s model, name of the output file in
#dot(graph) format using column of dataset
tree.export_graphviz(model, out_file="music-recommender.dot",
                     feature_names=["age", "gender"],
                     class_names=sorted(y.unique()),
                      label="all",
                      rounded=True,
                      filled=True)
```

In [ ]:
```
#Project using Django framework to build a web application
#reusable modules (building blocks)
```