MACHINE LEARNING

A machine learning project involves several steps.

Steps:

1. Importing Data
2. Clean the Data
3. Split the data into training & test sets
4. Create a Model
5. Train the Model
6. Make Predictions
7. Evaluate & Improve

Jupiter Shortcuts:

- python -m notebook / jupyter notebook (Command to open Jupiter notebook)
- Press 'Esc' (Escape Key) for Command mode (blue)/Edit mode (green)
- Press 'H' in command mode – For Keyboard shortcuts
- Press 'B' in command mode – It opens new cell below
- Press 'A' in command mode – It opens new cell Above
- Ctrl + Enter – Shortcut for Run
- Press 'Tab' key – For Intellisence
- Press 'Shift+Tab' on methos – It shows the details of that method
- 'Ctrl + /' – Shortcut comment/Uncomment both

The CSV file used in this tutorial:

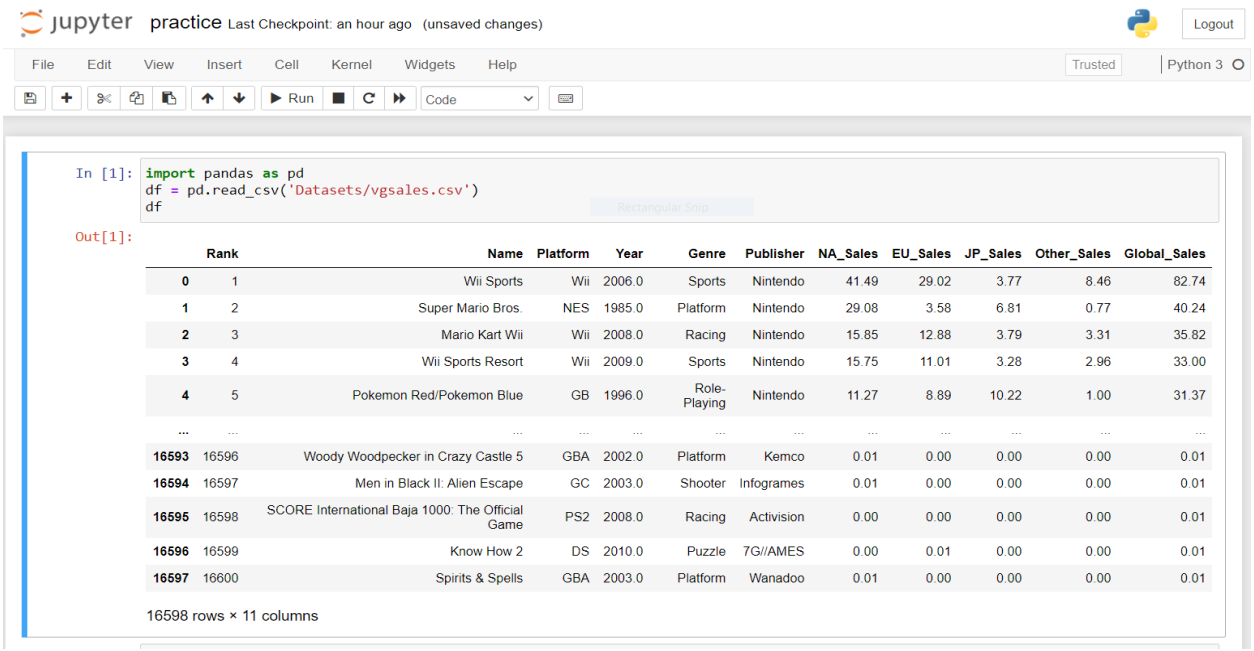
Music dataset - https://bit.ly/3muqqta

Video games dataset - https://www.kaggle.com/gregorut/videogamesales?select=vgsales.csv

1.Importing Data: Imports data from CSV file using ‘pandas’

Snippet:

```
import pandas as pd
df = pd.read_csv('Datasets/vgsales.csv')
df
```



jupyter practice Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [1]:
```
import pandas as pd
df = pd.read_csv('Datasets/vgsales.csv')
df
```

Out[1]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.77	8.46	82.74
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.79	3.31	35.82
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.28	2.96	33.00
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37
...	...	...	...	...	...	...	...	...	...	...	...
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00	0.00	0.00	0.01
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00	0.00	0.00	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00	0.00	0.00	0.01
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01	0.00	0.00	0.01
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.00	0.00	0.01

16598 rows × 11 columns

In [2]:
```
# Shape gives the size of dataset. (Rows, Columns)
df.shape
```

Out[2]: (16598, 11)

In [3]:
```
# describe gives information about each column in the dataset
df.describe()
```

Out[3]:

	Rank	Year	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales
count	16598.000000	16327.000000	16598.000000	16598.000000	16598.000000	16598.000000	16598.000000
mean	8300.605254	2006.406443	0.264667	0.146652	0.077782	0.048063	0.537441
std	4791.853933	5.828981	0.816683	0.505351	0.309291	0.188588	1.555028
min	1.000000	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000
25%	4151.250000	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000
50%	8300.500000	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000
75%	12449.750000	2010.000000	0.240000	0.110000	0.040000	0.040000	0.470000
max	16600.000000	2020.000000	41.490000	29.020000	10.220000	10.570000	82.740000

```
In [3]: # It gives data available in the dataset
        df.values

Out[3]: 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)
```

2. Clean the Data: If there are any junk/empty cells clean the data.

3. Split the data into training & test sets:

Split the data into both training & test. For getting accurate/best results we need to split the dataset into 80% as training and 20% test data.

```
In [5]: import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split

        music_data = pd.read_csv('Datasets/music.csv')
        X = music_data.drop(columns=['genre'])
        y = music_data['genre']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

4. Create a Model:

There are many models available. Here we are using '***DecisionTreeClassifier***'

```
In [5]: import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.model_selection import train_test_split

        music_data = pd.read_csv('Datasets/music.csv')
        X = music_data.drop(columns=['genre'])
        y = music_data['genre']
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
        model = DecisionTreeClassifier()
```

5. Train the Model:

Here we are training the model using ***X_train*** (input training data, which is 80%)**, *y_train*** (Output/target data)

```
In [10]: 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('Datasets/music.csv')
         X = music_data.drop(columns=['genre'])
         y = music_data['genre']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

         model = DecisionTreeClassifier()
         model.fit(X_train, y_train)  # Training a model
```

6. Make Predictions:

Here we are predicting using '***X_test'***, which is testing data (20%). It is different from trained data.

```
In [10]: 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('Datasets/music.csv')
         X = music_data.drop(columns=['genre'])
         y = music_data['genre']
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

         model = DecisionTreeClassifier()
         model.fit(X_train, y_train)
         predictions = model.predict(X_test)   # Predicting a trained model
```

7. Evaluate & Improve:

Below we can see the predicted accuracy, which is 1. So, it is 100% accurate.

In [11]:
```
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('Datasets/music.csv')
X = music_data.drop(columns=['genre'])
y = music_data['genre']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)

score = accuracy_score(y_test, predictions)  # Here we check the model accuracy.
# score = 1 is very good model
# score = 0 is poor model
score
# Here we got score as 1, it is a good model
```

Out[11]: 1.0

- If you change the test size to 80% ***('test_size=0.8')*** , see the prediction accuracy is very low (Here it is 0.2). So, divide the data as 80% for training and 20% for testing.
- If we train the model with more data, the accuracy will be more.

In [2]:
```
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('Datasets/music.csv')
X = music_data.drop(columns=['genre'])
y = music_data['genre']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8) # Testing with 80% and training the model with only 20%

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
predictions = model.predict(X_test)

score = accuracy_score(y_test, predictions)  # Here we check the model accuracy.
# score = 1 is very good model
# score = 0 is poor model
score
# Here we got score as 1, it is a good model
```

Out[2]: 0.2

In [ ]:

Note: If you run the model multiple times, accuracy will not be same. It may change every time.

Persisting/Reusing Models:

- Joblib object has methods for saving & loading models.
- Instead of creating & training a model every time, we can save the trained model using ***joblib*** and use it whenever required using ***load*** method.
- See below screen, model was saved as **'music-recommender.joblib'** using ***dump*** method.

```
In [4]: import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        import joblib

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

        model = DecisionTreeClassifier()
        model.fit(X, y)

        joblib.dump(model, 'music-recommender.joblib')

Out[4]: ['music-recommender.joblib']

In [ ]:
```

- In below snippet, we are loading the saved model using ***load*** method and predicted with input.

```
In [5]: import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        import joblib

        model = joblib.load('music-recommender.joblib')
        predictions = model.predict([[21,1]])
        predictions

Out[5]: array(['HipHop'], dtype=object)
```

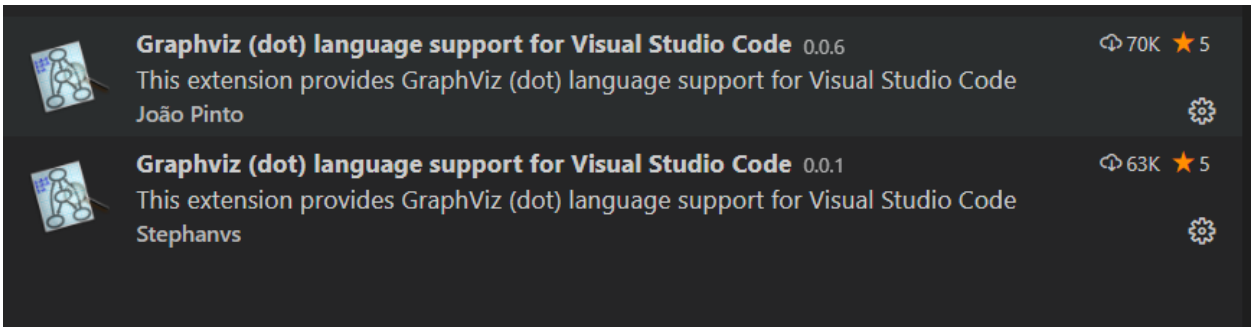
Visualizing a Decision tree:

Tree object has a method for exporting our decision tree in graphical format using below code.

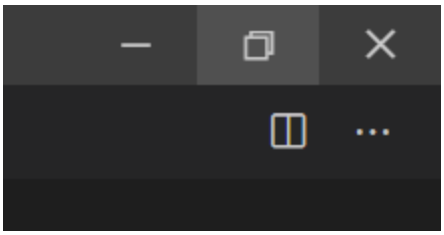
- Below code will be saved as '***music_recommnder.dot***'

```
In [6]: import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        from sklearn import tree

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

        model = DecisionTreeClassifier()
        model.fit(X, y)

        tree.export_graphviz(model, out_file='music_recommnder.dot',
                             feature_names=['age','gender'],
                             class_names=sorted(y.unique()),
                             label='all',
                             rounded=True,
                             filled=True)
```

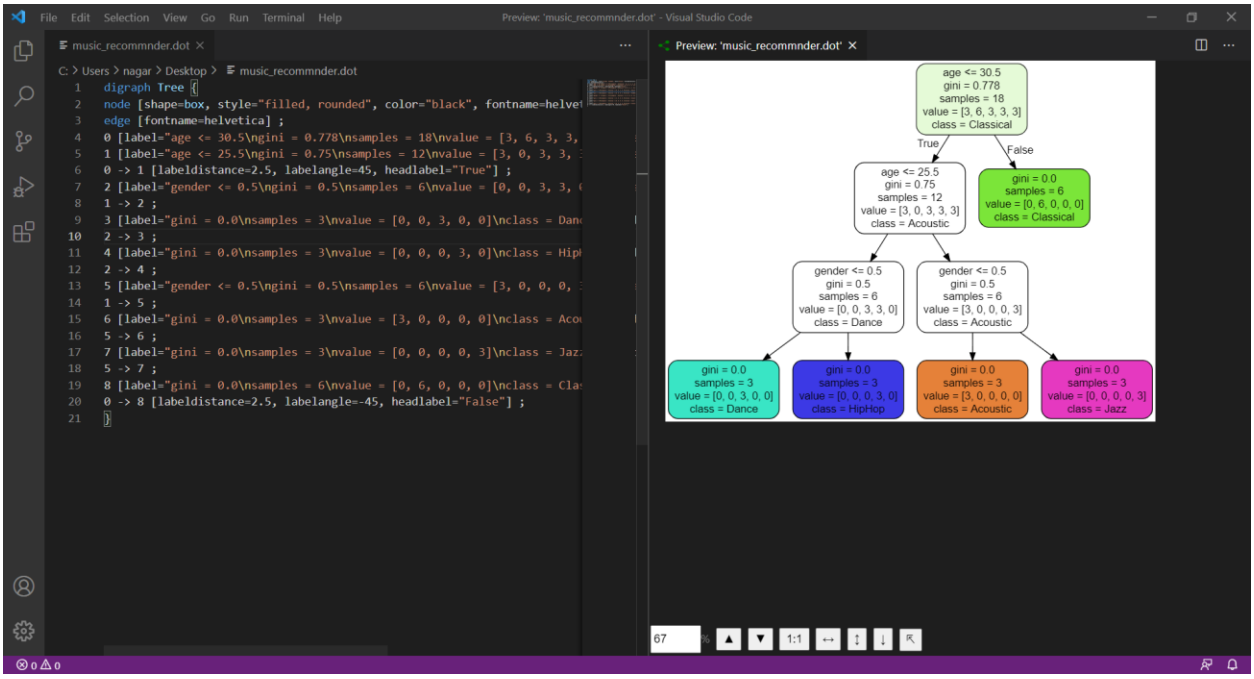
- Open that file (***music_recommnder.dot***) in vscode. Before that install any one of the below extensions (For me stephanvs didn't worked)



- Next open '`Open Preview to the Side`', it is available on the top right corner under 3 dots (…).



- Now, you can see the model visualization like below



Reference: https://www.youtube.com/watch?v=7eh4d6sabA0 (By Mosh Hamedani)