Video link: <https://www.youtube.com/watch?v=7eh4d6sabA0>

Machine Learning Steps:

1. Import the data
2. Clean the data (won’t need to do that, all of our data will be relevant)
3. Split the Data into Training/Testing sets
4. Create a Model
5. Train the Model
6. Make Predictions
7. Evaluate and Improve

Python Libraries and tools:

Libraries:

* Numpy: provides multi-dimensional array
* Pandas: provides a concept called data frame, a 2D data structure similar to a spreadsheet
* MatPlotLib: 2D plotting library for creating graphs and plots
* Scikit-Learn: provides common algorithms like decision trees, neural networks, and so on

**Jupyter**

We will be working in Jupytr to write our code, because other editors are not ideal for machine learning, because we will frequently need to inspect the data.

We will use Anaconda to download Jupyter. It will automatically download all of the popular libraries listed above.

**Starting Jupyter:**

Start the notebook server: jupyter notebook

File type of notebook: ipynb

Creating a new notebook will add the file on your computer

Pressing H shows Apple computer commands

You can delete a cell by pressing delete twice

df. And pressing tab shows methods and descriptions

Pressing ctrl and / converts a line into a comment

Ctrl + enter runs the current cell (so does clicking the cell, and clicking the run button)

**Kaggle**

Website containing data basees to test with

**Importing a data set** (video game sales):

import pandas as pd

df = pd.read\_csv(‘vgsales.csv’)

df.shape (shows how manny records and columns you have

Below that, in the second segment:

df.describe() gives basic statistics

Third segment:

df.values()

**Making predictions:**

In the example in the video, the following represents predicting the genre of music liked by 21 year old men, and 22 year old women:

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

music\_data = pd.read\_csv(‘music.cv’)

X = music\_data.drop(columns=[‘genre’])

y = music\_data[‘genre’]

model = DecisionTreeClassifier()

model.fit(X, y)

predictions = model.predict( [ [21 , 1], [22, 0] ] )

**Calculating the accuracy:**

80% of data should be for training, remaining 20% should be for testing.

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.model\_selection import train\_test\_split

music\_data = pd.read\_csv(‘music.cv’)

X = music\_data.drop(columns=[‘genre’])

y = music\_data[‘genre’]

X\_train, X\_test, y\_train, t\_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)

score

Will print out a score of up to 1.0 = 100% accurate

**Persisting Models:**

Joblib has methods for saving and loading models

This code will store the trained model in a joblib file:

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.externals import joblib

music\_data = pd.read\_csv(‘music.cv’)

X = music\_data.drop(columns=[‘genre’])

y = music\_data[‘genre’]

model = DecisionTreeClassifier()

model.fit(X, y)

jolib.dump(model, ‘music-recommender.joblib’)

# predictions = model.predict( [ [21 , 1], [22, 0] ] )

**Now, to load the model:**

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.externals import joblib

model = joblib.load(‘music-recommender.joblib’)

predictions = model.predict( [ [21 , 1], [22, 0] ] )

predictions

**Visualizing Decision Trees**

tree will allow us to display the tree in a graphical format:

import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn import tree

music\_data = pd.read\_csv(‘music.cv’)

X = music\_data.drop(columns=[‘genre’])

y = music\_data[‘genre’]

model = DecisionTreeClassifier()

model.fit(X, y)

tree.export\_graphiz(model, out\_file=’music-recommender.dot’,

feature\_names=[names[‘age’, ‘gender’], # so we can see the age and gender in each node

class\_name=sorted(y.unique()), # creates a unique list of genres, for displaying the class in each node

label=’all’, # adds readable labels

rounded=True,

filled=True) # fills each box with a color

Open the file in VS Code, and use the graphiz extension to view the graph.