About Me.....

- 1. Python Developer.
- 2. Working with Index-Data Project.
- 3. Keen interest in Open Source Contribtions (Specifically Scientific Data Analysis!)

What this Talk is About!

- How to run your Python code or Finance-Model on Quantum-computer, Without a Phd in Physics!
- What are possible ways to think about Quantum-Computing in terms of Application-Development.
- I am not a Quantum-Computing Advocate but would love to be one some day.....

About QML(Quantum Machine Learning!)

- Mix Classic Layers with Quantum Layers.
- Run 50% code in Classic Computer and rest either on Quantum-Computer Or Quantum-Simulator.
- Use the power of Q-bits!
- Think in terms of Physics(May be.... VQE)

Basics Of Quantum-Computing/Physics

Super-Position principle:

A Qubit/Electron/Photon could be at two places at once.

- Position of Each Qubit is based on probability.
- Classic-Bit is either 0 or 1
- |0>=[10] # Defineatly 0
- |1>=[01] # Defineatly 1

We have Quantum Gates!

$$\begin{array}{c} X \text{ Gate} \\ \text{Bit-flip, Not} \end{array} = \begin{array}{c} X \\ \end{array} = \begin{array}{c} \left[\begin{array}{c} 0 & 1 \\ 1 & 0 \end{array} \right] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{array}{c} \beta |0\rangle + \alpha |1\rangle \\ \end{array}$$

$$\begin{array}{c} Z \text{ Gate} \\ \text{Phase-flip} \end{array} = \begin{array}{c} Z \\ \end{array} = \begin{array}{c} \left[\begin{array}{c} 1 & 0 \\ 0 & -1 \end{array} \right] \begin{bmatrix} \alpha \\ \beta \end{array} = \begin{array}{c} \alpha |0\rangle - \beta |1\rangle \\ \end{array}$$

$$\begin{array}{c} H \text{ Gate} \\ \text{Hadamard} \end{array} = \begin{array}{c} H \\ \end{array} = \begin{array}{c} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{\alpha + \beta |0\rangle + \alpha - \beta |1\rangle}{\sqrt{2}} \\ \end{array}$$

$$\begin{array}{c} T \text{ Gate} \\ \end{array} = \begin{array}{c} T \\ \end{array} = \begin{array}{c} \left[\begin{array}{c} 1 & 0 \\ 0 & e^{i\pi/4} \end{array} \right] \begin{bmatrix} \alpha \\ \beta \end{array} = \begin{array}{c} \alpha |0\rangle + e^{i\pi/4} \beta |1\rangle \\ \end{array}$$

$$\begin{array}{c} Controlled \text{ Not} \\ Controlled X \\ CNot \end{array} = \begin{array}{c} \left[\begin{array}{c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array} \right] \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{array}{c} a|00\rangle + b|01\rangle + a|1\rangle \\ \end{array}$$

Swap
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \frac{a|00\rangle + c|01\rangle + b|10\rangle + d|11\rangle$$

5/12

Quantum-Entanglemet

- Spooky Action at distance!
- When we get two or more than two Qubits
 Entangled, measuring the results of one Qubit will let us know the state of other.
- A two part Documentry (Mind Bending)
 https://www.youtube.com/watch?v=ISdBAf-ysI0

How to run things on Quantum Computer?



Talk is cheap. Show me the code.

(Linus Torvalds)

IZQuotes

Set of Pythonic Libraries!

- Qiskit by IBM (https://qiskit.org/)
- Cirq by Google (https://quantumai.google/cirq)
- Amazon bracket
- And
- Pennylane! (My favorite one!)

Why Pennylane?

- Write Single set of Python-Code and Run it on each available Quantum-Computer/Simulator Which supports Pythonic-APIs.
- Supports Tensorflow, PyTorch, Numba, Jax as Quantum-Interfaces!
- Development is mostly targeted to Build Machine-learning Solutions!

Hello World Quantum-Computing!

```
import pennylane as qml
from pennylane import numpy as np
# create a quantum device
dev1 = qml.device("default.qubit", wires=1)
@gml.gnode(dev1)
def circuit(phi1, phi2):
  # a quantum node
  qml.RX(phi1, wires=0)
  qml.RY(phi2, wires=0)
  return qml.expval(qml.PauliZ(0))
def cost(x, y):
  # classical processing
  return np.sin(np.abs(circuit(x, y))) - 1
# calculate the gradient
dcost = qml.grad(cost, argnum=[0, 1])
                                               / 12
```

Different Quantum Computers Support!

- import pennylane a qml
- Dev1 = qml.device("cirq.qsim", wires=1, shots=100) # Runs on Google
- Dev2 = qml.device("qiskit.aer", wires=1, shots=100) # Runs on IBM
- Dev3 =
 qml.device("'microsoft.QuantumSimulator", wi
 res=1, shots=100) # Runs on Microsoft

QML in Action

- Activity:
 - 1. Create a Finance Model,
 - 2. Train it for T+100 Close-price Feature vector
 - 3. Get Results
 - 4. Add Quantum-layer and Re-Train!