Progress made:

The point cloud data was read into a jupyter notebook (create_dataset.ipynb) in order to analyze the data
The result was saved to a pickle file in the "data" folder (data.pkl)

In order to run the script,

1. Edit config.json
   Make changes to this file as needed, in terms of the number of epochs, learning rate, etc
2. In order to build the model, execute
   python main.py

If the name of the config file is not config.json, pass the name as a command-line argument to the script (eg "*python main.py config1*")

- The output of the model will be created in the experiment_data directory.
- A new folder will be created as per the name model name given in the config file
- During training, model state is saved in every epoch, so that in case model training aborts for some reason, we can resume from the last checkpoint
- The training and validation loss curves are also saved as a png file in the relevant experiment folder such that they're updated with every epoch

Details about the model:

Build a basic encoder-decoder architecture. Didn't use Voxelization and just passed the input points to the model as is.
The next step was to use Voxelization as a preprocessing step in order to scale the point cloud input but didn't get time to implement it.

Ran the experiment just for 2 epochs to verify that the code was working. Didn't get a chance to run thorough experiments using the model.

Visualization:

For visualizing the point cloud, started work on a util script that would make use of open3d visualize.
Sample command to execute: *python visualize.py model_name*
This script is not tested.

<u>Some issues that I faced during the task:</u>

1. Open3D pytorch doesn't work with Windows. Error: Open3D was not built with PyTorch support!
    a. In order to overcome this, used WSL with Ubuntu 20.04
    b. Created a new virtual environment in the mounted filesystem and used this setup to debug and experiment

2. Reinstalled pytorch using the specifications given in the Open3d git repo in order to resolve cuda related errors.

3. When running the visualization script, got an error that the system was unable to find "libEGL.so.1"
    Used the following to resolve it:

    For visualization: -->
    http://www.open3d.org/docs/latest/tutorial/visualization/headless_rendering.html
    EGL not found --> *sudo apt-install python3-pip mesa-utils libegl1-mesa xvfb*
    eglInitialize failed -->

```
git clone https://github.com/isl-org/Open3D
cd Open3D
util/install_deps_ubuntu.sh
mkdir build && cd build
cmake -DENABLE_HEADLESS_RENDERING=ON \
         -DBUILD_GUI=OFF \
         -DBUILD_WEBRTC=OFF \
         -DUSE_SYSTEM_GLEW=OFF \
         -DUSE_SYSTEM_GLFW=OFF \
         ..
   Error: CMake 3.19.2 or higher is required

   sudo apt install build-essential libtool autoconf unzip wget
   sudo apt purge --auto-remove cmake
   version=3.22
   build=2
   wget https://cmake.org/files/v$version/cmake-$version.$build.tar.gz
   tar -xzvf cmake-$version.$build.tar.gz
   cd cmake-$version.$build/
   ./bootstrap --> cmake c++ compiler doesn't support c++11 --> seems to be some
issue due to building on a mounted drive
      -- commented this check in CMakeLists.txt
      -- in case there's an issue finding openssl --> sudo apt-get install libssl-dev
   make -j$(nproc)
```

```
    Now rerun in the open3d buid folder: cmake
 -DENABLE_HEADLESS_RENDERING=ON \
           -DBUILD_GUI=OFF \
           -DBUILD_WEBRTC=OFF \
           -DUSE_SYSTEM_GLEW=OFF \
           -DUSE_SYSTEM_GLFW=OFF \
           ..
    make -j$(nproc)
    make install-pip-package
```

Using the above, I was able to import classes such as ml3d.vis.Visualizer(),
but this caused a version conflict for Open3d, due to which I got the same error as in the beginning
that is, "Open3D was not built with PyTorch support!"

This is a work in progress, I'm still working on resolving this issue.
I'd need some more time, but will be able to fix it.

References:

1. 3D Point Cloud Sematic Segmentation - *https://medium.com/analytics-vidhya/3d-point-cloud-semantic-segmentation-using-deep-learning-techniques-6c4504a97ce6#:~:text=Semantic%20segmentation%20is%20a%20technique,class%20as%20a%20single%20entity.&text=Given%20a%20point%20cloud%2C%20the,the%20semantic%20meanings%20of%20points.*
2. Submanifold Sparse Convolutions - *https://medium.com/geekculture/3d-sparse-sabmanifold-convolutions-eaa427b3a196*
3. Sparse Convolutions - *https://towardsdatascience.com/how-does-sparse-convolution-work-3257a0a8fd1*
4. Open3D-ML howtos - *https://github.com/isl-org/Open3D-ML/blob/master/docs/howtos.md*
5. Open3D Documentation - *http://www.open3d.org/docs/latest/python_api*
6. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks - *https://arxiv.org/pdf/1711.10275.pdf*
7. SECOND: Sparsely Embedded Convolutional Detection - *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6210968/pdf/sensors-18-03337.pdf*
8. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection - *https://arxiv.org/pdf/1711.06396.pdf*

9. Installation for Headless Rendering - *http://www.open3d.org/docs/latest/tutorial/visualization/headless_rendering.html*
10. Resolve cmake errors - *https://askubuntu.com/questions/355565/how-do-i-install-the-latest-version-of-cmake-from-the-command-line*