These instructions will explain the project up and running on your local machine for development and testing purposes.

Phase 1: Preprocessing the dataset using node embedding in the network using node2vec algorithm

Prerequisites

Importing the necessary libraries. Make sure you have tensorflow 2.0 backend and Keras interface in your environment.

```
#importing the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import csv
import itertools
%matplotlib inline
```

```
# Importing Machine Learning libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix as cm
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

#importing deep learning libraries
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.models import load_model
from keras.wrappers.scikit_learn import KerasClassifier
from keras.layers import Input, Convolution2D, MaxPooling2D, Dense, Dropout, Flatten
from keras.utils import np_utils
from keras.layers.merge import Concatenate
```

```
Using TensorFlow backend.
```

2.Reading and converting the shorter version of the concatenated ratings dataset into a dataframe. (shorter_network.csv)

3. For better and clean writing, making a function for the perception score features. The function is called GetNodefeatures to extract out the features of the node using the

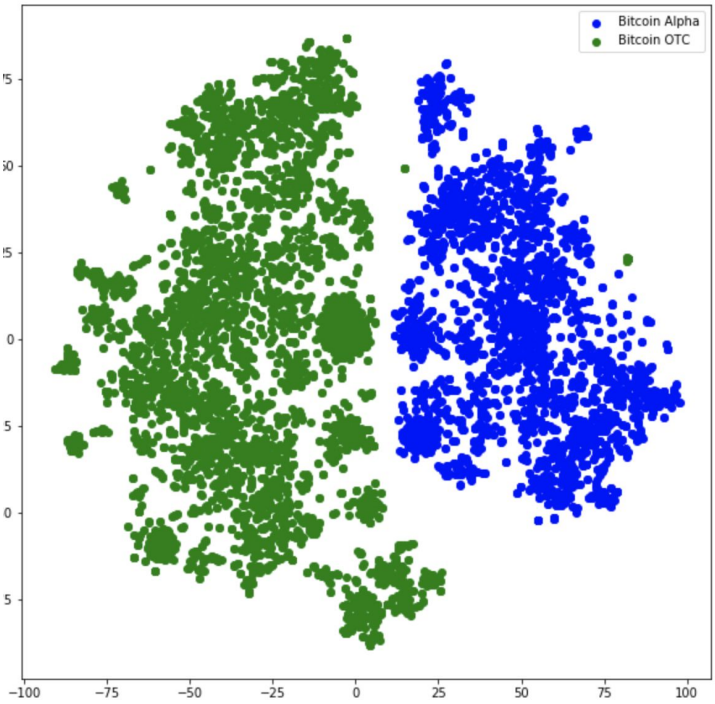
columns - “SOURCE” and “TARGET” of the dataset. Savings the node feature matrix file as a CSV file.

4.Running the node2vec implementation in python and storing it as the short_network.emb and then later creating a dictionary of the node and embeddings.

5. Reading the short_networknode_features.csv file and converting into a dataframe. Also, it is essential to create a network edgelist file and subsequently, we may create a dictionary of the nodes and embeddings to get an idea of the shape and dividing the node IDs for bitcoin Alpha and bitcoin OTC markets for a visualization later on.

6. Once, the edge list is created, it is important to create reduce the dimensions for a 2D projections on the embeddings using the predefined node dimensionality reduction algorithms like **t-SNE [1].**

7. Now that we have created a dictionary of the nodes, using that I divided the node IDs into the two different markets. Since the highest OTC node ID was 6005, it is inevitable that the other IDs will be bitcoin Alpha IDs.Then comes the scatter plot of exploratory data analysis.



8. Normalizing the features extracted from the node2vec algorithm and create a Node_Features_matrix file that contains the normalized values. After normalization, construct the input matrix and the output vector . Each row of the matrix contains 40 values (20 values for the source node or buyer, 20 values for the target node or seller). The vector captures the rating of the transaction the seller received from the buyer

NOTE: Here we assign a score of 1 if transaction is rated badly (fraud), and 0 otherwise

9. Calculating the mean, SD and norm versions of the Node_features and then capturing the normalized node ID in the dataframe (20 features from the range 1-21). Later, saving the file as a csv.

10. Creating the X input vector and the y output. Construct the input matrix X and the output vector y. Each row of the X matrix contains 40 values (20 values for the source node or buyer, 20 values for the target node or seller).The y vector captures the rating of the transaction the seller received from the buyer.

11. Repeating the same 10th step for the test dataset (validation set).

Phase 2: Model building, training and evaluation

Step 1: Creating training, testing dataset and splitting of the input matrix X and y Output into the 0.2 (test set) and 0.8 (train set) division. In addition to this, creating a function called ‘createsample’ function to generate samples of training data without biasing it towards the abundant category (honest ratings). Later, a concatenation of the fraud indices and not fraud indices lists created into the X_train and y_train sets.

Step 2. A generalized function for plotting a confusion matrix, as we need to check if the dataset is true and the classifier is predicting it correctly.

Step 3. Construct a Neural Network model of 6 layers with 4 hidden layers. Using Relu and Sigmoid as the activation functions. [2]

Later, using Nesterov adam and sgd optimizers to compile or train the model.

Step 4. Training the model using Bootstrapping. How is it done:

- Choose a number of bootstrap samples to perform
- Choose a sample size
- For each bootstrap sample
    - Draw a sample with replacement with the chosen size
    - Fit a model on the data sample
    - Estimate the skill of the model on the out-of-bag sample.
- Calculate the mean of the sample of model skill estimates.

Step 5. Repeating the same procedure above in the test set. Using the list (target_list) for honest and fraudulent values stored for later classification task. Lastly, plotting the normalized confusion matrix for the same.

Step 6: Repeating the steps 2-5 for a different models further : Logistic regressions using node2vec and perception scores, logistic regression with only perception scores, Neural network using both the features and MLP for binary classification

Phase 3 : Model evaluation and performance

Step 1: using the roc_auc_score() function of the sklearn metrics to plot all the different models that I tried out till now.

1. https://lvdmaaten.github.io/tsne/
2.https://keras.io/getting-started/sequential-model-guide/