1. Abstract
   1. A program that predicts the future of stock prices based upon its history using OHLC csv files.
2. Summary
   1. Temp
3. Data
   1. The data used for this project is a collection of both stocks and ETFs found [here](https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs).
   2. With the nature of ETFs being relatively stable, I will make this project focus more on the stock files in order to find an accurate model. ETFs will most likely be used to train a separate model due to the differences between the two
4. Research
   1. OHLC bars (Open, High, Low, Close)
      1. Definition
         1. Open: the price the stock started at for that period of time
         2. Close: the price the stock ended at for that period of time
         3. High: the highest point the price reached for that period of time
         4. Close: the lowest point the price reached for that period of time
      2. Important Info
         1. Each bar has a set period of time that it represents so this should be taken into account when comparing different datasets and looking at the overall success of each model.
         2. All four pieces of data may contribute to the overall success of the ML model however, they could also end up being redundant features that may need to be filtered out.
      3. Machine Learning Libraries and Models
         1. My original thought was that SKLearn could be used in order to predict on the future of stock prices however, since LSTM is the main solution for time-based problems this may not be the best library to use and I may instead move to using TensorFlow.
            1. **LSTM** refers to Long Short Term Memory and is typically used in cases where the history of data can significantly effect the future of the data. Especially with stocks this is why
         2. While **LSTM** is the main solution for these problems, I could still attempt a classification algorithm such as **Random Forests** or **Multi-Layer Perceptron** in order to see if there is any relation between the Open, High, and Low and where the Close is. This howeverdoes not have the power to predict further into the future and must predict on OCHL stick at a time.
         3. Since this fits the description of a regression problem more, a model such as **Gaussian Processes** could be used to predict the future of the stock prices as well. Similar to the classification methods, this will most likely be unable to predict far enough into the future to be completely however, it may be used a complimentary model in order to double check the predictions of our **LSTM** model.
      4. Conclusion
         1. The best approach will involve working with an LSTM model first to predict on the future of stocks. Once this approach is successful, we can either start to train a separate ETF model or add a complimentary Gaussian Processes model in order to hopefully improve accuracies slightly.
5. Process and Findings
   1. Model training on one file.
      1. This method is being done to see the degree of accuracy that can be reached on a single piece of data. The last 10-20% of stock variation will be used as the test set and compared to the model’s predictions based on the first 80-90% of the stock’s history. If this method can successfully train the model to a high percentage, then no further action may be needed. Most likely the model will need to be trained on the first 80-90% of multiple stocks before being capable of making accurate predictions.

# References

*Gaussian Processes*. (n.d.). Retrieved from scikit learn: https://scikit-learn.org/stable/modules/gaussian\_process.html

Mitchell, C. (2021, March 4). *Understanding an OHLC Chart and How to Interpret It*. Retrieved from Investopedia: https://www.investopedia.com/terms/o/ohlcchart.asp#:~:text=What%20is%20an%20OHLC%20Chart,most%20important%20by%20many%20traders.

*sklearn.ensemble.RandomForestClassifier*. (n.d.). Retrieved from scikit learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

*sklearn.neural\_network.MLPClassifier*. (n.d.). Retrieved from scikit learn: https://scikit-learn.org/stable/modules/generated/sklearn.neural\_network.MLPClassifier.html

TensorFLow. (n.d.). *tf.keras.layers.LSTM*. Retrieved from Tensorflow: https://www.tensorflow.org/api\_docs/python/tf/keras/layers/LSTM