Things to do:

* Find information about XGBOOST
  + **What:**
    - short for eXtreme Gradient Boosting, is an advanced machine learning algorithm. XGBoost is an ensemble learning algorithm that combines the predictions of multiple weak predictive models, called base learners, to create a strong predictive model. It belongs to the class of boosting algorithms, which iteratively trains weak models and assigns higher weights to misclassified instances to improve subsequent models
    - XGBoost operates by building a series of decision trees sequentially, where each subsequent tree corrects the errors made by the previous ones. It uses a gradient-based optimization technique to minimize an objective function, which is a combination of a loss function and a regularization term. The loss function quantifies the difference between predicted and actual values, while the regularization term controls the complexity of the model to prevent overfitting.
  + **When**
    - 2014
  + **By who**
    - initially developed by Tianqi Chen as part of his Ph.D. thesis at the University of Washington in 2014.
  + **Timelines**
    - Implementation of the gradient boosting framework by Friedman (2000, 2001)
  + **Initial languages**
    - Initially written in C++ but packages for XGBoost were made for R and Python soon after. Now, also have packages for Java, Scala, Julia, Perl, and other languages**. \*\*Need Years\*\***
  + **Citation**
  + **Company/major projects/ success stories**
    - This person tries to predict SP500 stocks using XG: <https://soulsinporto.medium.com/forecasting-sp500-stocks-with-xgboost-and-python-part-2-building-the-model-89c9c40e49fc>
    - Capital One, Chase, Bank of America, Amazon and more
* **List of educational videos and other resources**
  + Official XGBoost document that includes how to install and examples: https://xgboost.readthedocs.io/en/latest/?ref=nocode.ai
* **Links to downloads etc. (especially the python version)**
  + https://xgboost.readthedocs.io/en/latest/install.html#python
* <https://dl.acm.org/doi/10.1145/2939672.2939785>
* **Examples (find tutorials online)** 
  + Starter examples for R, Python, Julia, Scala: <https://xgboost.ai/about>
  + Maybe Dr. Li’s example in notes.
  + 4 part tutorial with regression and classification example and optimization (conceptual, no coding): https://www.youtube.com/playlist?list=PLblh5JKOoLULU0irPgs1SnKO6wqVjKUsQ
* **Optimization (hyperparameter tuning) for XGBOOST**
  + Different hyperparameters to tune based on what kind of problem: https://xgboost.readthedocs.io/en/latest/parameter.html
* **Optimization (hyperparameter tuning) for Random Forests**
  + https://scikit-learn.org/stable/modules/ensemble.html#random-forests-and-other-randomized-tree-ensembles
* **Comparison of the two optimized with Regression and repetitions (cross-validation)**
* Re-install anaconda, and all packages needed for Random forests (scikit-learn) and XGBOOST including Pandas, Matplotlib, Numpy and everything else and make a zoom video of it BUT, then save the video and upload it on YOUTUBE and then give me the link.

Sources:

* <https://ai-jobs.net/insights/xgboost-explained/>
* <https://en.wikipedia.org/wiki/XGBoost>
* <https://discovery.hgdata.com/product/xgboost>
* Dr. Li’s Notes

On GitHub, read me will have all this info and on the presentation will have this info. RF, XGBoost, Linear Regression: Compare all these methods in Python. Create all this info for RF.

Random Forest - presentation

* Find information about RANDOM FOREST
  + **What:**
    - Random forest is one of the most popular and powerful ensemble method used today in Machine Learning. An ensemble method consists of aggregating multiple outputs made by a diverse set of predictions, which in this case are decision trees. The purpose of this method is to average the outcome of individual predictions by diversifying the set of predictors, that is randomly choosing variables and observations for each tree. This will result in a high diversity and randomness and low variance, making forests less overfitting when compared to a single tree. Another attribute to randomforest is that it is very useful to determine variable importance, especially because of tree initialization.
    - Applications: Finance (High credit risk, detect fraud, pricing problems), healthcare (durg responses), e-commerce (engines for cross-sell)
    - Fun fact: The final outcome is an average in case of classification and the most voted class for classification. It's like each tree gets a say in what the forest believes!
  + **Timelines**
    - 1995 - Tin Kam Ho - Introduced the first algorithm for random decision forests.
    - 1996 - Leo Breiman - Introduced the concept of bagging (Bootstrap Aggregating)
    - Early 2000s - Leo Breiman and Adele Cutler - Combined Breiman's bagging idea with a random selection of features.
    - 2006 - "Random Forests" was registered as a trademark by Leo Breiman and Adele Cutler - registered "Random Forests" to Minitab.
    - Late 2000s to Early 2010s - Implementation of Random Forests in major programming languages R (randomForest) and Python (scikit-learn).
    - 2010s-Present: Random Forests have seen widespread adoption across various domains.