# Assignment

In this assignment you will use the Unsupervised Pre-Training + Supervised Fine-Tuning paradigm.

The objective is to wind up with a model that can perform Sentiment Analysis on sentences from a Financial domain.

Specifically

- you will choose a Pre-Trained Language Model from the HuggingFace platform
- Fine-Tune the model on the Financial PhraseBank dataset
    - a collection of hand-annotated sentences with sentiment classified a Negative, Neutral, Positive

The assignment will consist of

- a "basic" part
    - This is the minimum that you must submit (and succeed on) in order to earn a passing grade
- "extras"
    - more challenging objectives
    - earn points beyond the basic

# Financial PhraseBank dataset

The dataset was created as part of a research paper in Finance.

There were several humans who labeled the sentiment of sentences. Naturally humans don't always agree. Thus, the dataset has several "flavors" (called "subsets") based on the fraction of annotators who agree.

As part of the "basic" assignment, you should use the "all agree" flavor.

There are multiple ways to access this dataset.

- The preferred way: as a HuggingFace dataset (https://huggingface.co/datasets/financial_phrasebank)
- Discouraged: as a TFDS (https://www.tensorflow.org/datasets/community_catalog/huggingface/financial_pł
    - this is discouraged *only* because there will be an "extra" part asking you to c the HuggingFace dataset to a TFDS
    - you won't get credit for this "extra" if you obtain it directly as a TFDS !
- Discouraged: From the authors (https://www.researchgate.net/publication/251231364_FinancialPhraseBank-v10)
    - the "Download the PDF" link contains an archive file with the data
    - this is discouraged only because it involves extra work
    - but you might find it handy in creating your own TFDS in the "extra" part

You don't need to understand the paper but here is a reference (https://arxiv.org/pdf/1307.5336.pdf#page=9) to the paper that introduced it.

## Choose a Pre-Trained Language Model from the HuggingFace platform

There are lots of Language Models on HuggingFace. They vary by, among other things

- model size
- training objective
- language of training data

An important consideration is computational

- the model fits in the memory of the machine you are using
- the execution time is rapid enough for you to experiment
    - it will help to have a machine with a GPU (e.g., Google Colab)

You are free to choose *subject to limitations*

- the model should *not* have already been trained on a Financial dataset
    - e.g., you may not use FinBERT

I can verify that the choice of `distilbert-base-uncased` is feasible when using the free version of Google Colab

One of the options for extra credit is to experiment with different models and report and explain the differences.

## Submission requirements

You will submit a Jupyter notebook that follows the steps in the Recipe for Machine Learning (https://github.com/kenperry-public/ML_Fall_2022/blob/master/Recipe_Overview.ipynb)

That is, rather than just submitting a final model

- you should follow the steps (as appropriate) in the Recipe
- treat the notebook as a movie showing
    - the Recipe steps that you took
        - Exploratory Data analysis and Error Analysis are steps that are ignored at your own peril
    - including steps that led to failure (e.g., bad choices)
        - what you learned from the failure
        - how you overcame it or learned to make a different choice

Use Markup in Jupyter rather than code comments to convey your thoughts.

Use Section Headings to clearly indicate different parts of the assignment and different experiments.

Your code should be TensorFlow/Keras.

## Discussion

In each part, or each experiment, I would like to see a Discussion of results

- Can you conclude that your choices led to better of worse out of sample performance ?
- Do you have a theory as to why the performance changed/did not change ?

The objective is to get you to think like a scientist rather than just completing a task.

## Originality

There are no doubt published notebooks that use this dataset for a similar purpose.

The work that you submit should be your own.

## Getting started

The HuggingFace course is a great way to get up to speed on various aspects of this project.

## Basic part of assignment

There are several sub-parts.

- Fine-tune *only the Classifier head*
- Fine-tune *all* the weights

You will present a discussion of whether your out of sample performance was significantly improved by one of the two sub-parts.

The Basic part allows you several simplifications

- your model fitting can take features/labels that are Python data structures
    - these are most familiar to you
    - obtained from the dataset
- you can using a HuggingFace model that already comes with a Classifier head

I suggest that one section of your notebook clearly demonstrates success on the Basic Part.

- Walk before you try running
- Extra parts should be distinct suggestions

## Extras

You can demonstrate greater skill (and earn more points) by completing some extra parts.

After each extra, please complete a Discussion section as you did for the Basic part.

My suggestions follow, but I'm open to your ideas.

I will determine the amount of extra points by my perception of the difficulty of each extra.

## Create (and fit the model with) a TensorFlow Dataset (TFDS)

*Relative difficulty*: low to medium

The dataset is sufficiently small to fit into memory on a free Google Colab machine.

In this part, I want you to *pretend* that your machine is not big enough to fit all the data into memory.

You will create a TFDS and use this in the model fitting.

*Starting off with the HuggingFace dataset*, create a TFDS with the same data

- use this TFDS in the "fit" statement of the model
    - manipulate the data as much as possible using TFDS operations
        - e.g., if you need to shuffle the examples, use TFDS operations rather than shuffling before creating the TFDS
    - it's OK to load the entire HuggingFace dataset into memory for the purpose of creating the TFDS
- *you may not start off with a dataset that is already a TFDS*

The objective is for you to demonstrate skill with TFDS.

## Create your own Classification head

*Relative difficulty*: medium to hard

Rather than using a Classification head automatically provided with the HuggingFace model

- you will obtain a head-less model from HugginFace
- you will add your own Classification head
    - with as many layers as you like

The objective is for you to demonstrate some skills with the Keras Functional Model API.

## Use different ”flavors” of the dataset

*Relative difficulty*: low, but time-consuming

The basic part used examples on which all annotators agreed.

Try different flavors and discuss the results.

## Address any Imbalanced Data issues

*Relative difficulty: low*

- If the distribution between labels in the dataset is not uniform, you may want to address the imbalance

## Superior Error Analysis

*Relative difficulty: medium*

Rather than just reporting a single summary statistic for out of sample performance, analyze the results in detail

- Is one class harder to correctly classify than others
- Is there some systematic pattern of errors
    - e.g., characteristics of input sentences that are more difficult to correctly classify

Try to use the results of this analysis to improve the model

## Experiment with different Pre-Trained models

*Relative difficulty*: low, but time-consuming

Try several different pre-trained Language Models.

Discuss the results. For example

- does a bigger pre-trained model lead to better results
    - before and after fine-tuning all the weights
- does the type of data on which the model was trained make a difference

## Experiment with Fine-Tuning

*Relative difficulty*: low, but time-consuming

Does out of sample performance vary with changing

- the number of examples in Fine-Tuning
    - what is the smallest number that you think is sufficient
- there are many choices of proper subsets of a given size
    - does it matter which one you choose ?

## In-context learning

*Relative difficulty: low but fun !*

Can you use few-shot learning successfully (i.e., no further training) ?

It would be great to do this for Financial PhraseBank but the sentences may be too long

- pre-trained models have maximum sequence lengths that may be too small

Propose some interesting task related to Finance and try to achieve Few Shot Learning on the task.

```
In [2]: print("Done")
```

```
Done
```