Capitalized Title Here

by Jackson Moss, Author Two

Abstract An abstract of less than 150 words.

```
devtools::load_all()

#> Loading sandpr

library(dplyr)
library(tidyr)
library(keras)
library(tensorflow)
```

Introduction

The **sandpr** package provides data and functions that create stock portfolios for the equities market using machine learning methodology. It uses publically available data from Kaggle to train feed foward, fully connected nueral network models. The package provides functionality for both long only and market neutral (long short) portfolios for the calendar year of 2016.

Data

The raw data is in the form of two Kaggle datasets, which are read into R as csv files. The first dataset, `fundamentals_data`, contains yearly fundamental information for S&P 500 stocks. The column headers in this dataset are potential predictors for our neural network model, and include PE ratio, accounts receivable, depreciation, and many other information about company which the stock backs. The second dataset, `price_data` contains daily price information for S&P 500 stocks.

We load the datasets with the following.

```
price_data = read.csv("../data/prices.csv", header = T)
fundamentals_data = read.csv("../data/fundamentals.csv", header = T)
```

Cleaning these datasets is an important part of the analytical process. Generally, we need to subset for the correct stocks and the correct time periods, and then normalize the potential predictor variables. A function that does the above is as follows.

\begin{Schunk} \begin{Sinput} prepare_data = function(x, y) {

```
## Remove variables we don't want
x = x %>% select(-X, -For.Year)

## Coerce to correct classes
x$Ticker.Symbol = as.character(x$Ticker.Symbol)
y$symbol = as.character(y$symbol)

## Already have "%Y-%m-%d" in fundamentals data
y$date = as.Date(as.character(y$date), format = "%m/%d/%y")

## Convert date to date class
x$Period.Ending = as.Date(as.character(x$Period.Ending), format = "%Y-%m-%d")

## Remove information for anything after 2015
x = x %>% filter(Period.Ending <= "2015-12-31" & Period.Ending >= "2012-01-01")

## Only work with companies with price data that spans end of 2012 to end of 2016
## Get list of tickers that we want
companies = unique(y$symbol)
companies_keep = character()

lower_dates = as.Date(c("2013-12-31", "2013-12-30", "2013-12-29"), format = "%Y-%m-%d")
upper_dates = as.Date(c("2016-12-31", "2016-12-30", "2016-12-29"), format = "%Y-%m-%d")
```

```
## Loop through all companies
for(i in 1:length(companies)) {

    this_company = filter(y, symbol == companies[i])
    this_company_dates = this_company$date

    ## If we have the lower and upper range of our
   if(sum(lower_dates %in% this_company_dates) > 0 && sum(upper_dates %in% this_company_dates) > 0) {
       companies_keep = c(companies_keep, companies[i])
    }
}

## Subset for the companies we want
x = x %>% filter(Ticker.Symbol %in% companies_keep)

## Filter out all companies that do not contain fundamentals data
## for the date of "2015-12-31"
new_companies_keep = character()
possible_companies = unique(x$Ticker.Symbol)

for(i in 1:length(possible_companies)) {
    this_company_new = filter(x, Ticker.Symbol == possible_companies[i]) %>%
        select(Period.Ending)

    ## Check if we have date
    if(as.Date("2015-12-31") %in% this_company_new$Period.Ending) {
        new_companies_keep = c(new_companies_keep, possible_companies[i])
    }
}

## Subset for the companies we want
x = x %>% filter(Ticker.Symbol %in% new_companies_keep)

## Deal with missing values
## Set to average of dataset
columns = c("Cash.Ratio",
            "Current.Ratio",
            "Quick.Ratio",
            "Earnings.Per.Share",
            "Estimated.Shares.Outstanding")

for(i in 1:length(columns)) {
    x[[columns[i]]][is.na(x[[columns[i]]])] = mean(x[[columns[i]]], na.rm = TRUE)
}

## We should also normalize all fundamentals variables
## Get column names of numeric variables
numeric_names = character()

for(i in 1:(length(colnames(x)))) {

    if(is.numeric(x[, i])) {
        numeric_names = c(numeric_names, colnames(x[i]))
    }
}

## Scale the numeric columns
x <- x %>% mutate_each_(funs(scale(.) %>% as.vector), vars=numeric_names)

return(list("x" = x, "y" = y))
```

} \end{Sinput} \end{Schunk}

Next, we need to add a forward one year percentage price change variable to `fundamentals_data`. This includes looping through each stock/date combination in `fundamentals_data`, and searching

price_data for the price one year in the future. A function that does so is below.

\begin{Schunk} \begin{Sinput} process_data = function(x, y) {

```
## First, make sure our dates are in the same format
## Already have "%Y-%m-%d" in fundamentals data
y$date = as.Date(as.character(y$date), format = "%m/%d/%y")

changes = numeric()

## Loop through each row, get the date, then find the price one year in the future
for(i in 1:nrow(x)) {

    this_date = x[i, ]$Period.Ending
    this_symbol = x[i, ]$Ticker.Symbol

    ## Get the % change in price 1 year in the future
    price_now = filter(y, date == this_date, symbol == this_symbol) %>% select(close)
   price_future = filter(y, date == (this_date + 365), symbol == this_symbol) %>% select(close)

    ## This might not always work (weekends etc.)
    ## Keep trying the previous day until we get one that exists
    j = 1
    while(nrow(price_now) == 0) {

        ## Keep trying one day in the future
      price_now = filter(y, date == (this_date + j), symbol == this_symbol) %>% select(close)
        j = j + 1

        print(sprintf("Now Price %i row", i))
        print(sprintf("Now Price %i try", j))
    }

    k = 1
    while(nrow(price_future) == 0) {

        ## Keep trying one day in the past
      price_future = filter(y, date == (this_date + 365 - k), symbol == this_symbol) %>% select(close)
        k = k + 1

        print(sprintf("Future Price %i row", i))
        print(sprintf("Future Price %i try", k))
    }

## Calculate percentage change
percent_change = (price_future[1,1] - price_now[1,1]) / price_now[1,1]

## How to deal with stock splits
## Assume this means more than 50% decrease
## Let's just set to 0% change
if(percent_change <= -0.35) {
    percent_change = 0
}

changes = c(changes, percent_change)
}

## Append this to dataframe
x$one_year_price = changes

return(x)
```

} \end{Sinput} \end{Schunk}

Jackson Moss

Affiliation
line 1
line 2
author1@work

Author Two
Affiliation
line 1
line 2
author2@work