DS670 – SPRING 2018

Sabin Pradhan

1 Identifying the Problem Domain: What is the question?

The problem domain that this research focuses on is using Deep Learning to build a naive model for financial forecasting. The main question this research tries to answer is can linear factors models be outperformed by unburdening ourselves with the constraint of assuming the relation between the dependent variable and the independent variables are linear. This project will work to build a naive feed forward deep learning networks to predict the returns for mid cap financial sector stocks.

2 Literature Survey: What is known?

2.1 Artificial Neural Networks

Artificial Neural Networks is a machine learning architecture that was inspired by the operation of the biological brain. Much like how biological neurons process information in the brain, Artificial neural networks use mathematical units or artificial neurons to process information. These predictive models have been extremely successful in solving a wide variety of problems in the data science arena.

The scope of this project tried to study the relation between price movement, i.e. returns, in mid cap financial sector stocks, and their pricing and technical factors. It does not include the study of the time series aspect of the problem into the model building, thus is naive in that respect.

3 Tractability: Is there sufficient usable data?

Financial markets have been one of the most well documented industries in the twentieth century. For this project, the project references the End of Day US Stock Prices from Quandl.

End of Day US Stock Prices data set offers professional grade end-of-day adjusted and unadjusted prices for publicly-traded US stocks. The records in this datafile goes back to 1962 up until 2017. It compromises of the following 14 columns of data: 1)Ticker, 2) Date, 3) Unadjusted Open, 4) Unadjusted High, 5) Unadjusted Low, 6) Unadjusted Close, 7) Unadjusted Volume, 8) Dividends, 9) Splits, 10) Adjusted Open, 11) Adjusted High, 12) Adjusted Low, 13) Adjusted Close and 14) Adjusted Volume.

4 Design

The model for this project is going to be a 6 step process.

1. Data Extraction : The project needs to extract the financial sector mid cap stocks from the Quandl data set that contains all publicly traded US stock.

2. Calculating Returns : Most linear factor models utilize log returns since it normalizes the data. Here, since neural networks are a universal function approximator, the project will be working with adjusted close returns rather than log returns.

3. Calculating Technical Factors : The project then uses the End of Day pricing data of the financial sector mid-cap stocks to calculate the following technical factors: ADX, MACD,RSI, Stochastic Oscillators.

4. Feature Scaling: The project utilizes two methods to scale the data. Any data columns that are in the range [0,100] is divided by 100. These columns primarily include ADX and RSI. The other columns are scaled using the 40 day maximum absolute value. The scaled data is then exported from R into a csv file.

5. Model Building : The refined inputs are then going to be used to train a variety of feed forward neural networks with 3 hidden layers.

6. Model Testing : The model is then with the cross validation set during each epoch cycle then at the end is tested with a completely new unseen test data set for an unbiased performance review. If the model's performance in the cross validation set and the test set is comparable then we can be assured that the model generalizes well.

4.1 Tools

Python shall be used as the primary language for the data extraction/ model building part of this project. Python's TensorFlow library will be our tool of choice for building the deep learning networks. For the construction of technical factors and the feature scaling sections, this project will be utilizing R.

5 Extracting Relevant Data

The data set that Quandl provided had adjusted and unadjusted End of Day Prices to all US stocks. However, this project only focuses on mid-cap Financial sector stock prices. Thus, the data set needed to be filtered for the relevant data. The US Financial sector tickers could be downloaded from this link.The project uses Investopedia's definition of mid-cap as a company having market capitalization between $2 billion and $10 billion to narrow the data set.

The project uses the following python command to generate the Grep command to retrieve the tickers for mid cap financial companies.

Program in Python

```
import csv
tkr=[]
with open("Data/companylist.csv","r") as csvfile:
    csv_reader=csv.DictReader(csvfile, delimiter=',')
    for row in csv_reader:
        if float(row["MarketCap"])>=2*10**9 and
        ↪  float(row["MarketCap"])<=10*10**9:
            tkr.append(row["Symbol"])
```

The project then uses the newly constructed *tkr* list to construct a *grep* command to extract the mid-cap financial sector stocks from the End of Day pricing data.

Program in Python

```
command="grep -e '^"
for sym in range(len(tkr)):
    if sym==len(tkr)-1:
        command=command+" -e '^"+tkr[sym].strip()+"'"
    elif sym==0:
        command=command+tkr[sym].strip()+"'"
    else:
        command=command+" -e '^"+tkr[sym].strip()+"'"

command=command+" EOD_Date_Filtered.csv > eod_fin_mid_cap.csv"
```

Once the Grep command is completed, it's inserted into the Linux terminal to extract the financial sector mid-cap stock data and create the *eod_fin_mid_cap.csv* file.

6 Calculating Returns

Returns are calculated as the ratio of today's adjusted close to yesterday's adjusted close. To calculate this value, the data set is first imported into R as a data frame using the *read_csv* command. The program then renames the columns in the data set for better code readability as shown below.

Program in R:

```
eod=read.csv("eod_fin_mid_cap.csv")
colnames(eod)<-c("Ticker", "Date", "UnadjOpen", "UnadjHigh",
                 "UnadjLow", "UnadjClose", "UnadjVol", "Dividends",
                 "Splits", "AdjOpen", "AdjHigh", "AdjLow", "AdjClose",
                 "AdjVol")
```

The program then sorts the data by Ticker and date as shown below. In addition, the program also defines two variables *unique_tickers*, which is a set of uniques tickers in our data, and *eod_entries*, which returns the total number of observations in the data set. These variables will come in handy throughout the next sections.

Program in R:

```
eod<-eod[order(eod$Ticker,eod$Date),]
unique_tickers<-unique(eod$Ticker)
eod_entries<-dim(eod)[1]
```

Finally, the program calculates the Returns as follows:

Program in R:

```
#########################################################
# Returns
#########################################################

returns<-vector()
prev_ticker=""
for(i in 1:eod_entries){
  if(eod$Ticker[i]!=prev_ticker){
    returns=c(returns, NA)
  }
  else{
    returns[i]<-(eod$AdjClose[i]/eod$AdjClose[i-1])
  }
  prev_ticker=eod$Ticker[i]
}

eod$returns_adj_close<-returns
```

7 Constructing the Technical Factors

As mentioned before, this project takes uses four technical factors, i.e., ADX, MACD, RSI, and Stochastic Oscillators. The indicators were constructed in R.

7.1 ADX

The Average Directional Index(ADX) is a technical indicator used to gauge the strength of a trend. This is a non directional indicator, thus quantifies a trend's strength regardless of the direction of price movement. There are a lot of pieces involved in constructing ADX and shall go through each one in this section.

Up-Move Up-move is calculated as the difference between today's adjusted high and yesterdays adjusted low. In contrast to up-move, down-move is the difference between yesterday's adjusted high to today's adjusted low. The program calculates the up-move and down-move for each of the stocks on any given day as follows:

Program in R:

```
upmove<-vector()
downmove<-vector()
prev_ticker=""
for(i in 1:eod_entries){
  if(eod$Ticker[i]!=prev_ticker){
    upmove[i]=NA
    downmove[i]=NA
  }
  else{
    upmove[i]<-eod$AdjHigh[i]-eod$AdjHigh[i-1]
    downmove[i]<-eod$AdjLow[i-1]-eod$AdjLow[i]
  }
  prev_ticker=eod$Ticker[i]
}
eod$upmove<-upmove
eod$downmove<-downmove
```

+DM and -DM +DM or Plus Directional Movement is the up-move if the up-move is greater than the down-move and greater than zero for a stock in any given day else +DM is equal to zero. In contrast to +DM, -DM or Minus Directional Movement is the down-move if the down-move is greater than the up-move and greater than zero for a stock in any given day else -DM is equal to zero.

Program in R:

```
pos.DM<-vector()
neg.DM<-vector()
prev_ticker=""
for(i in 1:eod_entries){
  if(eod$Ticker[i]!=prev_ticker){
    pos.DM[i]<-NA
```

```
    neg.DM[i]<-NA
  }
  else{

    if(eod$upmove[i]>eod$downmove[i] & eod$upmove[i]>0){
      pos.DM[i]<-eod$upmove[i]
      neg.DM[i]<-0
    }
    else if(eod$downmove[i]>eod$upmove[i] & eod$downmove[i]>0){
      neg.DM[i]<-eod$downmove[i]
      pos.DM[i]<-0
    }
    else{
      pos.DM[i]<-0
      neg.DM[i]<-0
    }
  }
  prev_ticker<-eod$Ticker[i]
}
eod$pos.DM<-pos.DM
eod$neg.DM<-neg.DM
```

True Range True Range for a stock on any given day is defined as the greatest of the following figures:

1. Difference between today's adjusted high and today's adjusted low
2. Absolute difference between today's adjusted high and yesterday's adjusted close
3. Absolute difference between today's adjusted low and yesterday's adjusted close.

The code for this is given below:

Program in R:

```
tr<-vector()
prev_ticker=""
for(i in 1:eod_entries){
  if(eod$Ticker[i]!=prev_ticker){
    tr[i]<-NA
  }
  else{
    tr[i]<-max((eod$AdjHigh[i]-eod$AdjLow[i]),
               abs(eod$AdjHigh[i]-eod$AdjClose[i-1]),
               abs(eod$AdjLow[i]-eod$AdjClose[i-1]))
  }
  prev_ticker<-eod$Ticker[i]
}
eod$tr<-tr
```

+DI and -DI +DI or Plus Directional Indicator is the ratio of the 14 day simple moving averages of +DM and the true range. -DI or Minus Directional Indicator is the ratio of the 14 day simple moving averages of -DM and the true range.

Program in R:

```
#14day
pos.DI=vector()
neg.DI=vector()

for(i in 1:length(unique_tickers)){
  temp.df=eod[eod$Ticker==unique_tickers[i],c("pos.DM","neg.DM","tr")]
  pos.DI<-c(pos.DI, (100*movavg(temp.df$pos.DM, 14, type="s"))/
  movavg(temp.df$tr, 14, type="s"))
  neg.DI<-c(neg.DI, (100*movavg(temp.df$neg.DM, 14, type="s"))/
  movavg(temp.df$tr, 14, type="s"))
}
eod$pos.DI<-pos.DI
eod$neg.DI<-neg.DI
```

Once all the above indicators are in place, the ADX for 5-, 14-,30- and 60-days can be calculated as using the following formula:

$$ADX = 100 \times \frac{SMA[(+DI) - (-DI)]}{(+DI) + (-DI)}$$

Program in R:

```
#########################################################
#ADX
#########################################################
adx.5=vector()
adx.14=vector()
adx.30=vector()
adx.60=vector()

for(i in 1:length(unique_tickers)){
  temp.df=eod[eod$Ticker==unique_tickers[i],c("pos.DI","neg.DI")]

  #5 day
  adx.5<-c(adx.5, (100*movavg(abs(temp.df$pos.DI-temp.df$neg.DI),5,type="s"))/
  (temp.df$pos.DI+temp.df$neg.DI))

  #14 day
  adx.14<-c(adx.14, (100*movavg(abs(temp.df$pos.DI-temp.df$neg.DI),14,type="s"))/
  (temp.df$pos.DI+temp.df$neg.DI))

  #30 day
  adx.30<-c(adx.30, (100*movavg(abs(temp.df$pos.DI-temp.df$neg.DI),30,type="s"))/
  (temp.df$pos.DI+temp.df$neg.DI))
```

```
  #60 day
  adx.60<-c(adx.60, (100*movavg(abs(temp.df$pos.DI-temp.df$neg.DI),60,type="s"))/
  (temp.df$pos.DI+temp.df$neg.DI))
}

eod$adx.5<-adx.5
eod$adx.14<-adx.14
eod$adx.30<-adx.30
eod$adx.60<-adx.60
```

7.2 RSI

The Relative Strength Index (RSI) is a technical indicator that gauge the momentum of any given stock. This indicator compares the magnitude of gains and losses over a given time period, 14 days in this case, to measure if any given stock is overbought or oversold. Like ADX, RSI has a few pieces that need to be put in place before it can be computed.

Gain and Loss If today's adjusted closing price is greater than yesterday's adjusted closing price, then Gain is calculated as the greater difference between those numbers, else it is equal to zero. If yesterday's adjusted closing price is greater than today's adjusted closing price then Loss is calculated as the greater difference between those numbers, else it is equal to zero.The code for this is shown below:

Program in R:

```
gain<-vector()
loss<-vector()
prev_ticker<-""
for(i in 1:eod_entries){
  if(eod$Ticker[i]!=prev_ticker){
    gain[i]=NA
    loss[i]=NA
  }
  else{
    if(eod$AdjClose[i]-eod$AdjClose[i-1]>0){
      gain[i]=eod$AdjClose[i]-eod$AdjClose[i-1]
      loss[i]=0
    }
    else{
      gain[i]=0
      loss[i]=eod$AdjClose[i-1]-eod$AdjClose[i]

    }
  }
  prev_ticker=eod$Ticker[i]
}
eod$gain<-gain
eod$loss<-loss
```

Average Gain and Loss Average Gain and Average Loss is a 14 day simple moving average of gain and loss values for any given stock.

Program in R:

```
#Average Gain and Loss
avg.gain=vector()
avg.loss=vector()
for(i in 1:length(unique_tickers)){
  tic_df = eod[eod$Ticker==unique_tickers[i],c("gain","loss")]
  avg.gain=c(avg.gain, movavg(tic_df$gain,14,type="s"))
  avg.loss=c(avg.loss, movavg(tic_df$loss,14,type="s"))
}

eod$Avg.Gain<-avg.gain
eod$Avg.Loss<-avg.loss
```

RS The Relative Strength of any given stock is the ratio of the two week simple moving averages of gain and loss.

Program in R:

```
#RS
eod$rs<-eod$Avg.Gain/eod$Avg.Loss
```

Average RS The Average Relative Strength of a stock is calculated as the two week simple moving average of the RS.

Program in R:

```
ma.rs<-vector()
for(i in 1:length(unique_tickers)){
  rs = eod[eod$Ticker==unique_tickers[i],c("rs")]
  ma.rs<-c(ma.rs, movavg(rs,14,type="s"))
}
eod$ma.rs<-ma.rs
```

Once Average RS has been calculated, RSI can be calculated using the following formula:

$$RSI = 100 - \frac{1}{100 + SMA[RS]}$$

7.3 MACD

The Moving Average Convergence Divergence is a technical indicator that utilizes short term and long term trends to anticipate price movements on the underlying stock. It is another momentum indicator that show the relationship between two moving averages of price. The calculations for this technical indicator is fairly straight forward. The program looks at MACD for 5vs10, 10vs20, 20vs40, and 40vs80 day. It does this by first calculating the exponential moving adjusted closing

price averages for the specified time period as follows:

Program in R:

```
#MACD
ma.5<-vector()
ma.10<-vector()
ma.20<-vector()
ma.40<-vector()
ma.80<-vector()

for(i in 1:length(unique_tickers)){
  AdjClose = eod[eod$Ticker==unique_tickers[i],c("AdjClose")]
  cat(unique_tickers[i],length(AdjClose),"\n")
  ma.5<-c(ma.5, movavg(AdjClose,5,type="e"))
  ma.10<-c(ma.10, movavg(AdjClose,10,type="e"))
  ma.20<-c(ma.20, movavg(AdjClose,20,type="e"))
  ma.40<-c(ma.40, movavg(AdjClose,40,type="e"))
  ma.80<-c(ma.80, movavg(AdjClose,80,type="e"))
}

eod$ma.5<-ma.5
eod$ma.10<-ma.10
eod$ma.20<-ma.20
eod$ma.40<-ma.40
eod$ma.80<-ma.80
```

Once the exponential moving averages have been computed, it is only a matter of taking differences between longer term moving averages and shorter term moving averages as shown below.

Program in R:

```
eod$macd.10v5<-eod$ma.10-eod$ma.5
eod$macd.20v10<-eod$ma.20-eod$ma.10
eod$macd.40v20<-eod$ma.40-eod$ma.20
eod$macd.80v40<-eod$ma.80-eod$ma.40
```

7.4 Stochastic Oscillators

Stochastic Oscillators is a technical indicator that compares the adjusted close to the range of price fluctuations over any given period of time. The program computes this momentum indicator for 5,14, 30 and 60 day time period. The code is given below.

Program in R:

```
#########################################################
# Stochastic Oscillators
#########################################################
```

```
percent.K.5<-vector()
percent.K.14<-vector()
percent.K.30<-vector()
percent.K.60<-vector()

for(i in 1:length(unique_tickers)){
  temp.df = eod[eod$Ticker==unique_tickers[i],c("AdjClose","AdjHigh","AdjLow")]
  #5 day
  K.5<-(temp.df$AdjClose-rollapply(temp.df$AdjLow, width=5, FUN=min, fill=NA,
   ↪  align="right"))/
    (rollapply(temp.df$AdjHigh, width=5, FUN=max, fill=NA, align="right")-
       rollapply(temp.df$AdjLow, width=5, FUN=min, fill=NA, align="right"))
  percent.K.5<-c(percent.K.5, K.5)

  #14 day
  K.14<-(temp.df$AdjClose-rollapply(temp.df$AdjLow, width=14, FUN=min, fill=NA,
   ↪  align="right"))/
    (rollapply(temp.df$AdjHigh, width=14, FUN=max, fill=NA, align="right")-
       rollapply(temp.df$AdjLow, width=14, FUN=min, fill=NA, align="right"))
  percent.K.14<-c(percent.K.14, K.14)

  #30 day
  K.30<-(temp.df$AdjClose-rollapply(temp.df$AdjLow, width=30, FUN=min, fill=NA,
   ↪  align="right"))/
    (rollapply(temp.df$AdjHigh, width=30, FUN=max, fill=NA, align="right")-
       rollapply(temp.df$AdjLow, width=30, FUN=min, fill=NA, align="right"))
  percent.K.30<-c(percent.K.30, K.30)

  #60 day
  K.60<-(temp.df$AdjClose-rollapply(temp.df$AdjLow, width=60, FUN=min, fill=NA,
   ↪  align="right"))/
    (rollapply(temp.df$AdjHigh, width=60, FUN=max, fill=NA, align="right")-
       rollapply(temp.df$AdjLow, width=60, FUN=min, fill=NA, align="right"))
  percent.K.60<-c(percent.K.60, K.60)


}

eod$percent.K.5<-percent.K.5
eod$percent.K.14<-percent.K.14
eod$percent.K.30<-percent.K.30
eod$percent.K.60<-percent.K.60
```

8 Feature Scaling

The data needs to be standardize before being used to train the model. Before moving on to the normalization techniques, it is important to go over the normalization factors.

8.1 40 Day Maximum Absolute Value

All pricing data i.e. Adjusted Open, Adjusted High,Adjusted Low and Adjusted Close is scales with respect the the 40 day maximum absolute value of the Adjusted closing price. The program uses the 40 day maximum absolute value of the Adjusted closing price to scale all the adjusted closing price moving averages as well. Each of the MACD calculations and the %K stochastic oscillators also uses their respective 40 day maximum absolute value to scale their numbers.

Program in R:

```
#Calculating 40 day Maximum Absolute Value
max.vol<-vector()
max.close<-vector()
max.macd.10v5<-vector()
max.macd.20v10<-vector()
max.macd.40v20<-vector()
max.macd.80v40<-vector()
max.percent.K.5<-vector()
max.percent.K.14<-vector()
max.percent.K.30<-vector()
max.percent.K.60<-vector()



for(i in 1:length(unique_tickers)){
  temp.df=eod[eod$Ticker==unique_tickers[i],]

  max.vol<-c(max.vol,rollapply(abs(temp.df$AdjVol),
  width=40,FUN=max,fill=NA, align="right"))

  max.close<-c(max.close, rollapply(abs(temp.df$AdjClose),
  width=40,FUN=max,fill=NA, align="right"))

  max.macd.10v5<-c(max.macd.10v5, rollapply(abs(temp.df$macd.10v5),
  width=40,FUN=max,fill=NA, align="right"))

  max.macd.20v10<-c(max.macd.20v10, rollapply(abs(temp.df$macd.20v10),
  width=40,FUN=max,fill=NA, align="right"))

  max.macd.40v20<-c(max.macd.40v20, rollapply(abs(temp.df$macd.40v20),
  width=40,FUN=max,fill=NA, align="right"))

  max.macd.80v40<-c(max.macd.80v40, rollapply(abs(temp.df$macd.80v40),
  width=40,FUN=max,fill=NA, align="right"))
```

```
  max.percent.K.5<-c(max.percent.K.5, rollapply(abs(temp.df$percent.K.5),
  width=40,FUN=max,fill=NA, align="right"))

  max.percent.K.14<-c(max.percent.K.14, rollapply(abs(temp.df$percent.K.14),
  width=40,FUN=max,fill=NA, align="right"))

  max.percent.K.30<-c(max.percent.K.30, rollapply(abs(temp.df$percent.K.30),
  width=40,FUN=max,fill=NA, align="right"))

  max.percent.K.60<-c(max.percent.K.60, rollapply(abs(temp.df$percent.K.60),
  width=40,FUN=max,fill=NA, align="right"))
}

#Normalizing Data
eod$AdjVol<-eod$AdjVol/max.vol
eod$AdjClose<-eod$AdjClose/max.close
eod$AdjOpen<-eod$AdjOpen/max.close
eod$AdjHigh<-eod$AdjHigh/max.close
eod$AdjLow<-eod$AdjLow/max.close
eod$ma.5<-eod$ma.5/max.close
eod$ma.10<-eod$ma.10/max.close
eod$ma.20<-eod$ma.20/max.close
eod$ma.40<-eod$ma.40/max.close
eod$ma.80<-eod$ma.80/max.close
eod$macd.10v5<-eod$macd.10v5/max.macd.10v5
eod$macd.20v10<-eod$macd.20v10/max.macd.20v10
eod$macd.40v20<-eod$macd.40v20/max.macd.40v20
eod$macd.80v40<-eod$macd.80v40/max.macd.80v40

eod$percent.K.5<-eod$percent.K.5/max.percent.K.5
eod$percent.K.14<-eod$percent.K.14/max.percent.K.14
eod$percent.K.30<-eod$percent.K.30/max.percent.K.30
eod$percent.K.60<-eod$percent.K.60/max.percent.K.60
```

8.2 ADX and RSI Scaling

Since ADX and RSI are numbers between 0 and 100, the program divides the values by 100 to scale it between 0 and 1.

Program in R:

```
eod$rsi<-eod$rsi/100
eod$adx.5<-eod$adx.5/100
eod$adx.14<-eod$adx.14/100
eod$adx.30<-eod$adx.30/100
eod$adx.60<-eod$adx.60/100
```

8.3 Data Export

Before the program exported the data into a new csv file, the data had a couple rows and columns that need to be omitted before the project could move forwards with the model building process. First was the unadjusted pricing data that was omitted from the dataframe, and then were 11 observations that had infinity values due to a consecutive 14 day or more zero value for the +DI and -DI column which resulted in an unusual spike in ADX.

```
#Dropping UnAdj Pricing Data
eod<-eod[,-c(3:9)]
View(eod)
#Omitting na from dataframe and saving dataframe as myeod.csv
eod1=na.omit(eod[-c(66217,66218,155291,369084,369085,369086,
77609,477610,477611,477612,624549),-c(24,25)])
View(eod1)
write.csv(eod1,file = "myeod.csv", row.names = FALSE)
```

9 Fitting the Model: Training

The project builds a 3 layered deep learning network using the python TensorFlow Library. There is three steps to building a model in TensorFlow.

First, the program imports the data into a pandas data frame and splits the dataset into training, cross validation set and testing set in the ratio 6:2:2.

The program then defines an acyclic operation map. During this stage the program conceptually builds the model schematics, and defines a cost function and optimizer. The cost function lets the program define how to assess the performance of the model while the optimizer defines a method to tune the weights and biases so that cost can be minimized.

Lastly, the program needs to define a session that runs the operation map defined earlier. In this stage of the model building process, the session feeds the filtered data set into the previously defined operation map and tunes the variables to minimize the given cost function using the specified optimizer.

9.1 Data Set Up

The program imports the data set into python as follows:

Program in Python:

```
#load data
df=pd.read_csv("myeod.csv")
df.dropna(axis=0, how='all')
df.head()
```

In addition to importing the data set, the code above also removes any values that may be NAs.

The program then does a 60:40 split of the entire data set into the training and cross validation/ testing set. The cross validation/ testing set is further splits into cross validation and testing set in a 1:1 ratio. The code is shown below:

Program in Python:

```
X_train, X_cv_test, Y_train, Y_cv_test = sk.train_test_split(
    df.drop(["Ticker","Date","log_returns_adj_close",
    "AdjVol","rs","ma.rs"].astype(np.float32), axis=1),
    df["log_returns_adj_close"].astype(np.float32),
    test_size=0.40,
    train_size =0.60,
    random_state = 42)

X_cv, X_test, Y_cv, Y_test = sk.train_test_split(
    X_test,
    Y_test,
    test_size=0.50,
    random_state = 42)
```

9.2 Hyperparameters

The model building starts by first defining the hyper parameters variables *training_epoch* and *learning_rate*. *training_epoch* defines how many times should program go through the entire training data to train the model. *learning_rate* sets the step size for the optimizer.

Program in Python:

```
learning_rate = 0.001
training_epochs = 150
```

9.3 Operation Map

First the program specifies the number of inputs and the number of outputs.

Program in Python:

```
number_of_inputs = 35
number_of_output = 1
```

The program then specify the number of nodes for each layer.

Program in Python:

```
layer_1_nodes = 20
layer_2_nodes = 150
layer_3_nodes = 10
```

Now, the program defines a placeholder for the input matrix. The *shape* parameter is defined as (*None*, *number_of_inputs*). Here, *None* specifies a fluid dimension for the input matrix. After that the program does onto define a scope for the variable for better hierarchy.

Program in Python:

```
# Input Layer
with tf.variable_scope('input'):
    X = tf.placeholder(tf.float32, shape=(None, number_of_inputs))
```

The program then proceeds to define the first layer. the program uses the *get_variable* function to define *weights* and *biases*. For *weights*, the program needs to specify it's *shape* and initializes it using the *xavier_initializer*(). It then designates the *biases* vector and initializes it to zero. The program then multiplies the X variable defined above with the *weights* variable and adds *biases* to it. It thens passes it though a sigmoid functions and designates it to *layer_1_output*.

Program in Python:

```
with tf.variable_scopevariable ('layer_1'):
    weights = tf.get_variable(name = "weights1",
            shape = [number_of_inputs, layer_1_nodes],
        initializer=tf.contrib.layers.xavier_initializer())
    biases = tf.get_variable(name = "biases1",
            shape = [layer_1_nodes],
            initializer = tf.zeros_initializer())
    layer_1_output = tf.nn.sigmoid( tf.matmul(X, weights) + biases)
```

The program defines hidden layer 2 and 3 in the same manner. The only think that is different is the shape of the *weightes* and *biases* variable depending on the layer that comes before it and the layer that comes after it.

Program in Python:

```
#Layer 2
with tf.variable_scopevariable ('layer_2'):
    weights = tf.get_variable(name = "weights2",
            shape = [layer_1_nodes, layer_2_nodes],
            initializer=tf.contrib.layers.xavier_initializer())
    biases = tf.get_variable(name = "biases2",
            shape = [layer_2_nodes],
            initializer = tf.zeros_initializer())
    layer_2_output = tf.nn.sigmoid( tf.matmul(layer_1_output, weights) + biases)

#Layer 3
with tf.variable_scope('layer_3'):
    weights = tf.get_variable(name = "weights3",
            shape = [layer_2_nodes, layer_3_nodes],
            initializer=tf.contrib.layers.xavier_initializer())
    biases = tf.get_variable(name = "biases3",
            shape = [layer_3_nodes],
            initializer = tf.zeros_initializer())
    layer_3_output = tf.nn.sigmoid( tf.matmul(layer_2_output, weights) + biases)
```

The output layer is defined a little bit differently. The program defines *weights* and *biases* similar to layer 1, 2 and 3. However, since the output layer is a regression, it does not use a sigmoid function.

Program in Python:

```
with tf.variable_scope('output'):
    weights = tf.get_variable(name = "weights4",
            shape = [layer_3_nodes, number_of_output],
            initializer=tf.contrib.layers.xavier_initializer())
    biases = tf.get_variable(name = "biases4",
            shape = [number_of_output],
            initializer = tf.zeros_initializer())
    output_layer = tf.matmul(layer_3_output, weights) + biases
```

Now, the program defines the cost function. However, since the cost function require the target variable, the program defines a placeholder Y, which is then used to calculate the mean squared error cost function.

Program in Python:

```
with tf.variable_scope('cost'):
    Y = tf.placeholder(tf.float32, shape=(None, 1))
    cost = tf.reduce_mean(tf.squared_difference(output_layer,Y))
```

Once the cost function is defined, the program defines an optimizer that uses an Adam optimizer, which is an extension of the stochastic gradient descent method, from the TensorFlow module which will be later used to minimize the cost function. **Program in Python:**

```
with tf.variable_scope('train'):
    optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

One all of the above components are defined, we create a summary variable that helps us log our findings and display them on TensorBoard.

Program in Python:

```
with tf.variable_scope("logging"):
    tf.summary.scalar("current_cost",cost)
    summary=tf.summary.merge_all()
```

9.4 Running Sessions

The program first creates a *Sessions*() object *session*. It then goes on to initialize all the variables it created earlier by running the *global_variables_initializer* function through the *run session* method.

Once all the variables have been initialized, the program then creates the summary writers *training_writer* and *cv_writer* which will helps log the finding and later connect them to TensorBoard.

After, the program loops through each predefined *training_epoch* and minimizes the cost function by tuning the weights and biases using the *optimizer* defined in the earlier section. It achieves this by passing the *optimizer* through the *run Session* method and using the $feed\_dict$ parameter to pass the X_train and Y_train data.

At each fifth loop, the program retrieves the *cost* and *summary* variables for both the training and cross validation set using the *run Session* method and writes it to the predefined log file location. In addition, it also prints the cost value for both the training and cross validation set for the loop and gives users real time feedback to see if the model is being trained properly.

After the program loops though all the epoch cycles, it print out the final training, cross validation and test cost before ending.This values will be available via TensorBoard, which allows for an interactive local web interface to study the model.

Program in Python:

```
print ("Layer 1 nodes: ", layer_1_nodes)
print ("Layer 2 nodes: ", layer_2_nodes)
print ("Layer 3 nodes: ", layer_3_nodes)

print("\n=================================================================\
=========\n=================================================================\
===========\n")

with tf.Session() as session:

    # Run the global variable initializer to initialize all variables and layers
    ↪  of the neural network
```

```
session.run(tf.global_variables_initializer())

# Create log file writers to record training progress.
# We'll store training and testing log data separately.
training_writer = tf.summary.FileWriter('./logs/200/training-'+
str(layer_1_nodes)+'-'+str(layer_2_nodes)+'-'+str(layer_3_nodes),
session.graph)

cv_writer = tf.summary.FileWriter('./logs/200/cv-'+
str(layer_1_nodes)+'-'+str(layer_2_nodes)+'-'+str(layer_3_nodes),
session.graph)

# Run the optimizer over and over to train the network.
# One epoch is one full run through the training data set.
for epoch in range(training_epochs):

    # Feed in the training data and do one step of neural network training
    session.run(optimizer, feed_dict={X: X_train, Y: Y_train})

    # Every 5 training steps, log our progress
    if epoch % 5 == 0:
        # Get the current accuracy scores by running the "cost" operation on
        ↪  the training and test data sets
        training_cost, training_summary = session.run([cost, summary],
        feed_dict={X: X_train, Y:Y_train})
        cv_cost, cv_summary = session.run([cost, summary],
        feed_dict={X: X_cv, Y: Y_cv})

        # Write the current training status to the log files (Which we can
        ↪  view with TensorBoard)
        training_writer.add_summary(training_summary, epoch)
        cv_writer.add_summary(cv_summary, epoch)

        # Print the current training status to the screen
        print("Epoch {}: \t Training Cost: {}  \t CV Cost:
        ↪  {}".format(epoch,training_cost,cv_cost))

# Training is now complete!

# Get the final accuracy scores by running the "cost" operation on the
↪  training and test data sets
final_training_cost = session.run(cost, feed_dict={X: X_train, Y: Y_train})
final_cv_cost = session.run(cost, feed_dict={X: X_cv, Y: Y_cv})
final_testing_cost=session.run(cost, feed_dict={X: X_test, Y: Y_test})
print("\n=================================================================\
=========\n=================================================================\
===========\n")
print("Final Training cost:\t {}".format(final_training_cost))
```

```
print("Final CV cost      :\t {}".format(final_cv_cost))
print("Final Testing cost :\t {}".format(final_testing_cost))
```

10 Performance

The mean squared error for a variety of the models built are given below. The table has been arranged by ascending mean squared test error. The training set had 415705 observations while the cross validation set and testing set had 138568 and 138569 observations respectively. The 3 layered feed forward neural network had best performance on all training, cross validation and testing data sets. The full cost output for each epoch for each model is given in the appendix.

Note: Below, training cost is usually higher than the testing and cross validation cost value, however this is due to the relatively high volume of observations in the training set compared to the test and cross validation set.

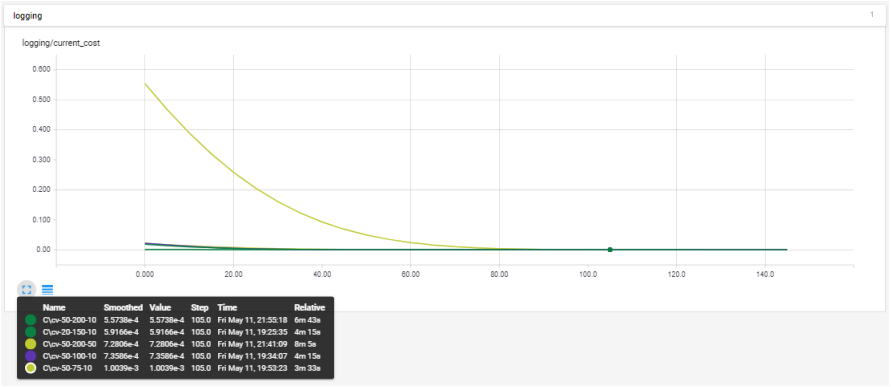
Table 1: Performance of 3 Layered Feed Forward Neural Networks

Model_ID	Hidden Layers	Training MSE	CV MSE	Testing MSE
1	[50,200,10]	0.00056610518	0.00054402207	0.00054071512
2	[50,200,50]	0.00059777679	0.00057585532	0.00057141017
3	[20,150,10]	0.00060730847	0.00058444921	0.00058087759
4	[50,100,10]	0.00071898662	0.00069601752	0.00069170940
5	[50,75,10]	0.00088104373	0.00086086511	0.00085606932
6	[300,200,10]	0.01235671621	0.01233010925	0.01231437549
7	[15,150,15]	0.05187492445	0.05183893070	0.05181110650
8	[5 ,150,10]	0.56025499105	0.56021565199	0.56013315916
9	[20,100,10]	0.61855840683	0.61846590042	0.61837440729
10	[20,200,10]	1.39943194389	1.39930152893	1.39916813374
11	[20,15,10]	7.12661552429	7.12637138367	7.12583684921

From the table above, the 3 layered neural network with layers [50, 200, 100], i.e., Model Id 1, performs the best out of all the models. It has the lowest mean squared error cost value for all training (0.00056610518), cross validation (0.00054402207) and test set (0.00054071512). Not to mention, the test and cross validation cost values are comparable which indicates that the model generalizes well.

Model 2 and model 3 have comparable training, cross validation and testing cost values of (0.00059777679, 0.00060730847), (0.00057585532, 0.00058444921) and (0.00057141017, 0.00058087759) respectively. Both models generalized well.

The cost output for each epoch cycle for the top 5 models is given below via Tensorboard's interactive visualizations.:



11 Limitations and Improvements

In the model built here for the most part ignores the time series aspect of the data set. Including that aspect of the data into the model building process would be a tremendous improvement and would generate better insights.

Instead of using a fully connected neural networks, a recurrent neural network would be able to utilize the time series nature of the data set and generate deeper insights. Even better, to overcome the vanishing gradient problem of recurrent neural networks, would be to use a Long-Short Term Memory (LSTM) neural network which would be able to follow trends over much longer horizons. This could improve the mean squared errors by a great margin.

In addition, incorporation fundamental factors to the data set would give the data more valuable information. In addition, to condense the number of fundamental and technical factors, the project could employ Stacked Autoencoders to reduce the dimension and generate deeper insights.

12 Appendix

Below is the full performance output for each of the model for each epoch:

12.1 Model ID: 1

Layer 1 nodes: 50
Layer 2 nodes: 200
Layer 3 nodes: 10

==========
==========
Epoch 0: Training Cost: 0.001581798424012959 CV Cost: 0.0015618331963196397
Epoch 5: Training Cost: 0.0014321224298328161 CV Cost: 0.0014154112432152033
Epoch 10: Training Cost: 0.0012882787268608809 CV Cost: 0.001270294189453125
Epoch 15: Training Cost: 0.0012000263668596745 CV Cost: 0.0011805443791672587
Epoch 20: Training Cost: 0.001097215572372079 CV Cost: 0.0010778839932754636
Epoch 25: Training Cost: 0.0010188270825892687 CV Cost: 0.0009998125024139881
Epoch 30: Training Cost: 0.0009415801614522934 CV Cost: 0.0009220116771757603
Epoch 35: Training Cost: 0.0008795488392934203 CV Cost: 0.0008593837264925241
Epoch 40: Training Cost: 0.0008223836775869131 CV Cost: 0.0008021561079658568
Epoch 45: Training Cost: 0.0007760632433928549 CV Cost: 0.0007557316566817462
Epoch 50: Training Cost: 0.000735535635612905 CV Cost: 0.000714848167262971 4
Epoch 55: Training Cost: 0.0007024820661172271 CV Cost: 0.0006815724191255867
Epoch 60: Training Cost: 0.0006749873282387853 CV Cost: 0.0006539768073707819
Epoch 65: Training Cost: 0.000652501592412591 CV Cost: 0.0006313021876849234
Epoch 70: Training Cost: 0.0006344739813357592 CV Cost: 0.0006130996625870466
Epoch 75: Training Cost: 0.0006199587369337678 CV Cost: 0.0005984845920465887
Epoch 80: Training Cost: 0.0006084908964112401 CV Cost: 0.0005869006854481995
Epoch 85: Training Cost: 0.0005994884995743632 CV Cost: 0.0005777839687652886
Epoch 90: Training Cost: 0.0005924340803176165 CV Cost: 0.0005706564406864345
Epoch 95: Training Cost: 0.0005869477754458785 CV Cost: 0.000565099238883704
Epoch 100: Training Cost: 0.0005826805718243122 CV Cost: 0.0005607650964520872
Epoch 105: Training Cost: 0.000579337531235069 CV Cost: 0.0005573765956796706
Epoch 110: Training Cost: 0.0005766939138993621 CV Cost: 0.0005546944448724389
Epoch 115: Training Cost: 0.0005745699163526297 CV Cost: 0.000552539131604135
Epoch 120: Training Cost: 0.0005728251417167485 CV Cost: 0.0005507703172042966
Epoch 125: Training Cost: 0.0005713507416658103 CV Cost: 0.0005492785712704062
Epoch 130: Training Cost: 0.0005700666224583983 CV Cost: 0.0005479844403453171
Epoch 135: Training Cost: 0.0005689154495485127 CV Cost: 0.0005468300078064203
Epoch 140: Training Cost: 0.0005678576417267323 CV Cost: 0.000545769406016916
Epoch 145: Training Cost: 0.0005668659578077495 CV Cost: 0.0005447794683277607

==========
==========
Final Training cost: 0.0005661051836796105
Final CV cost : 0.0005440220702439547
Final Testing cost : 0.0005407151184044778

12.2 Model ID: 2

Layer 1 nodes: 50
Layer 2 nodes: 200
Layer 3 nodes: 50

==========
==========

Epoch 0: Training Cost: 0.5538824200630188 CV Cost: 0.5538293719291687
Epoch 5: Training Cost: 0.4678104817867279 CV Cost: 0.4677591323852539
Epoch 10: Training Cost: 0.3896690905094147 CV Cost: 0.38962024450302124
Epoch 15: Training Cost: 0.319854199886322 CV Cost: 0.3198108375072479
Epoch 20: Training Cost: 0.2585318684577942 CV Cost: 0.25849059224128723
Epoch 25: Training Cost: 0.20560941100120544 CV Cost: 0.2055712342262268
Epoch 30: Training Cost: 0.16076615452766418 CV Cost: 0.16073110699653625
Epoch 35: Training Cost: 0.1234825849533081 CV Cost: 0.12344859540462494
Epoch 40: Training Cost: 0.09308778494596481 CV Cost: 0.09305576235055923
Epoch 45: Training Cost: 0.06881000101566315 CV Cost: 0.06877975910902023
Epoch 50: Training Cost: 0.04982893913984299 CV Cost: 0.049800291657447815
Epoch 55: Training Cost: 0.03531863912940025 CV Cost: 0.03529089689254761
Epoch 60: Training Cost: 0.024485453963279724 CV Cost: 0.024458860978484154
Epoch 65: Training Cost: 0.016598371788859367 CV Cost: 0.016572749242186546
Epoch 70: Training Cost: 0.011008019559085369 CV Cost: 0.010983130894601345
Epoch 75: Training Cost: 0.007157844491302967 CV Cost: 0.0071336012333631516
Epoch 80: Training Cost: 0.004587389528751373 CV Cost: 0.004563664551824331
Epoch 85: Training Cost: 0.0029286134522408247 CV Cost: 0.0029053206089884043
Epoch 90: Training Cost: 0.0018978200387209654 CV Cost: 0.0018748610746115446
Epoch 95: Training Cost: 0.0012383985679037869 CV Cost: 0.0012613090220838785
Epoch 100: Training Cost: 0.0009361054981127381 CV Cost: 0.0009136499720625579
Epoch 105: Training Cost: 0.0007503464003093541 CV Cost: 0.0007280579302459955
Epoch 110: Training Cost: 0.0006583500653505325 CV Cost: 0.0006361822015605867
Epoch 115: Training Cost: 0.0006172387511469424 CV Cost: 0.0005951642524451017
Epoch 120: Training Cost: 0.0006015769322402775 CV Cost: 0.0005795665201731026
Epoch 125: Training Cost: 0.000597241974901408 CV Cost: 0.0005752708530053496
Epoch 130: Training Cost: 0.000597030739299953 CV Cost: 0.0005750843556597829
Epoch 135: Training Cost: 0.0005977167165838182 CV Cost: 0.0005757880280725658
Epoch 140: Training Cost: 0.0005981359863653779 CV Cost: 0.0005762168439105153
Epoch 145: Training Cost: 0.0005980822024866939 CV Cost: 0.0005761597421951592

==========
==========

Final Training cost: 0.0005977767868898809
Final CV cost : 0.0005758553161285818
Final Testing cost : 0.000571410171687603

12.3 Model ID: 3

Layer 1 nodes: 20
Layer 2 nodes: 150
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 0.01915532536804676 CV Cost: 0.019123703241348267
Epoch 5: Training Cost: 0.014483371749520302 CV Cost: 0.014452873729169369
Epoch 10: Training Cost: 0.010567880235612392 CV Cost: 0.010538537055253983
Epoch 15: Training Cost: 0.007417359855026007 CV Cost: 0.00738913007080555
Epoch 20: Training Cost: 0.005001192446798086 CV Cost: 0.0049740043468773365
Epoch 25: Training Cost: 0.0032506268471479416 CV Cost: 0.0032244303729385138
Epoch 30: Training Cost: 0.0020658220164477825 CV Cost: 0.0020405028481036425
Epoch 35: Training Cost: 0.0013280475977808237 CV Cost: 0.001303493743762374
Epoch 40: Training Cost: 0.0009149890975095332 CV Cost: 0.0008910759352147579
Epoch 45: Training Cost: 0.0007152145844884217 CV Cost: 0.0006918067228980362
Epoch 50: Training Cost: 0.0006388602196238935 CV Cost: 0.0006158282631076872
Epoch 55: Training Cost: 0.0006222465308383107 CV Cost: 0.0005994679522700608
Epoch 60: Training Cost: 0.000626626773737371 CV Cost: 0.0006040094303898513
Epoch 65: Training Cost: 0.0006328807794488966 CV Cost: 0.000610336777754128
Epoch 70: Training Cost: 0.0006346108275465667 CV Cost: 0.0006120855687186122
Epoch 75: Training Cost: 0.0006319743115454912 CV Cost: 0.000609427341260016
Epoch 80: Training Cost: 0.0006272793398238719 CV Cost: 0.0006046851049177349
Epoch 85: Training Cost: 0.0006226132391020656 CV Cost: 0.000599966268055141
Epoch 90: Training Cost: 0.0006190563435666263 CV Cost: 0.0005963526200503111
Epoch 95: Training Cost: 0.0006167530082166195 CV Cost: 0.0005940051632933319
Epoch 100: Training Cost: 0.0006153791327960789 CV Cost: 0.0005925885634496808
Epoch 105: Training Cost: 0.0006144812796264887 CV Cost: 0.0005916640511713922
Epoch 110: Training Cost: 0.0006137279560789466 CV Cost: 0.0005908949533477426
Epoch 115: Training Cost: 0.0006129654939286411 CV Cost: 0.0005901223630644381
Epoch 120: Training Cost: 0.000612148956861347 CV Cost: 0.000589300121646374 5
Epoch 125: Training Cost: 0.0006112967967055738 CV Cost: 0.0005884449928998947
Epoch 130: Training Cost: 0.0006104418425820768 CV Cost: 0.0005875880597159266
Epoch 135: Training Cost: 0.0006095985881984234 CV Cost: 0.0005867439904250205
Epoch 140: Training Cost: 0.000608770817052573 CV Cost: 0.000585917208809405 6
Epoch 145: Training Cost: 0.0006079549202695489 CV Cost: 0.0005850986926816404

==========
==========

Final Training cost: 0.0006073084659874439
Final CV cost : 0.0005844492116011679
Final Testing cost : 0.0005808775895275176

12.4 Model ID: 4

Layer 1 nodes: 50
Layer 2 nodes: 100
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 0.02226964756846428 CV Cost: 0.022251108661293983
Epoch 5: Training Cost: 0.01674705184996128 CV Cost: 0.016727730631828308
Epoch 10: Training Cost: 0.012144285254180431 CV Cost: 0.012124392203986645
Epoch 15: Training Cost: 0.00846762116998434 CV Cost: 0.008447092026472092
Epoch 20: Training Cost: 0.00567457964643836 CV Cost: 0.0056534418836236
Epoch 25: Training Cost: 0.0036760587245225906 CV Cost: 0.0036543849855661392
Epoch 30: Training Cost: 0.002345630433410406 CV Cost: 0.0023234591353684664
Epoch 35: Training Cost: 0.0015355685027316213 CV Cost: 0.001512986491434276
Epoch 40: Training Cost: 0.0010960492072626948 CV Cost: 0.0010731180664151907
Epoch 45: Training Cost: 0.0008931765332818031 CV Cost: 0.0008699854370206594
Epoch 50: Training Cost: 0.0008215627167373896 CV Cost: 0.0007981779053807259
Epoch 55: Training Cost: 0.0008089469047263265 CV Cost: 0.0007854469586163759
Epoch 60: Training Cost: 0.0008134943782351911 CV Cost: 0.0007899399497546256
Epoch 65: Training Cost: 0.0008162775193341076 CV Cost: 0.0007927161059342325
Epoch 70: Training Cost: 0.0008124519372358918 CV Cost: 0.0007889115950092673
Epoch 75: Training Cost: 0.0008037034422159195 CV Cost: 0.0007802078616805375
Epoch 80: Training Cost: 0.0007932816515676677 CV Cost: 0.0007698385743424296
Epoch 85: Training Cost: 0.0007835981668904424 CV Cost: 0.0007602131809107959
Epoch 90: Training Cost: 0.0007756173727102578 CV Cost: 0.0007522813975811005
Epoch 95: Training Cost: 0.0007692360668443143 CV Cost: 0.0007459534681402147
Epoch 100: Training Cost: 0.0007639073301106691 CV Cost: 0.0007406665245071054
Epoch 105: Training Cost: 0.0007590631139464676 CV Cost: 0.0007358584553003311
Epoch 110: Training Cost: 0.0007543485262431204 CV Cost: 0.0007311750086955726
Epoch 115: Training Cost: 0.0007496310281567276 CV Cost: 0.000726487603969872
Epoch 120: Training Cost: 0.0007449085242114961 CV Cost: 0.0007217891397885978
Epoch 125: Training Cost: 0.0007402252522297204 CV Cost: 0.0007171278703026474
Epoch 130: Training Cost: 0.0007356191636063159 CV Cost: 0.0007125466363504529
Epoch 135: Training Cost: 0.0007311059162020683 CV Cost: 0.0007080630166456103
Epoch 140: Training Cost: 0.0007266959873959422 CV Cost: 0.0007036745664663613
Epoch 145: Training Cost: 0.0007223770953714848 CV Cost: 0.0006993855931796134

==========
==========

Final Training cost: 0.0007189866155385971
Final CV cost : 0.0006960175232961774
Final Testing cost : 0.0006917093996889889

12.5 Model ID: 5

Layer 1 nodes: 50
Layer 2 nodes: 75
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 0.021441202610731125 CV Cost: 0.021441824734210968
Epoch 5: Training Cost: 0.017247026786208153 CV Cost: 0.017245376482605934
Epoch 10: Training Cost: 0.013594388030469418 CV Cost: 0.013590460643172264
Epoch 15: Training Cost: 0.010498713701963425 CV Cost: 0.010492577217519283
Epoch 20: Training Cost: 0.00795357208698988 CV Cost: 0.007945439778268337
Epoch 25: Training Cost: 0.005930406507104635 CV Cost: 0.005920396186411381
Epoch 30: Training Cost: 0.004381061997264624 CV Cost: 0.004369319416582584
Epoch 35: Training Cost: 0.003242408623918891 CV Cost: 0.0032291323877871037
Epoch 40: Training Cost: 0.002442803466692567 CV Cost: 0.002428158652037382
Epoch 45: Training Cost: 0.0019086432876065373 CV Cost: 0.001892849220894277
Epoch 50: Training Cost: 0.0015707040438428521 CV Cost: 0.0015539494343101978
Epoch 55: Training Cost: 0.0013687944738194346 CV Cost: 0.0013512545265257359
Epoch 60: Training Cost: 0.0012545305071398616 CV Cost: 0.0012363579589873552
Epoch 65: Training Cost: 0.0011920391116291285 CV Cost: 0.0011733915889635682
Epoch 70: Training Cost: 0.0011569606140255928 CV Cost: 0.0011379644274711609
Epoch 75: Training Cost: 0.001134306425228715 CV Cost: 0.001115066115744412
Epoch 80: Training Cost: 0.0011159591376781464 CV Cost: 0.0010965463006868958
Epoch 85: Training Cost: 0.0010982215171679854 CV Cost: 0.0010787013452500105
Epoch 90: Training Cost: 0.0010799533920362592 CV Cost: 0.0010603525443002582
Epoch 95: Training Cost: 0.0010611678007990122 CV Cost: 0.0010415192227810621
Epoch 100: Training Cost: 0.0010422716150060296 CV Cost: 0.001022582408040762
Epoch 105: Training Cost: 0.0010236307280138135 CV Cost: 0.0010039068292826414

Epoch 110: Training Cost: 0.0010054782032966614 CV Cost: 0.0009857176337391138
Epoch 115: Training Cost: 0.0009878823766484857 CV Cost: 0.0009680872899480164
Epoch 120: Training Cost: 0.0009708469151519239 CV Cost: 0.0009510066010989249
Epoch 125: Training Cost: 0.0009543189080432057 CV Cost: 0.000934427313040942
Epoch 130: Training Cost: 0.0009382454445585608 CV Cost: 0.0009183013462461531
Epoch 135: Training Cost: 0.0009226112742908299 CV Cost: 0.0009026059997268021
Epoch 140: Training Cost: 0.000907389388885349 CV Cost: 0.000887325790245086
Epoch 145: Training Cost: 0.000892586715053767 CV Cost: 0.0008724590879864991

==========
==========

Final Training cost: 0.000881043728441
Final CV cost : 0.0008608651114627719
Final Testing cost : 0.0008560693240724504

12.6 Model ID: 6

Layer 1 nodes: 300
Layer 2 nodes: 200
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 0.5105771422386169 CV Cost: 0.5105352997779846
Epoch 5: Training Cost: 0.4539581537246704 CV Cost: 0.45391884446144104
Epoch 10: Training Cost: 0.40128934383392334 CV Cost: 0.4012509882450104
Epoch 15: Training Cost: 0.3529181480407715 CV Cost: 0.35288000106811523
Epoch 20: Training Cost: 0.30904144048690796 CV Cost: 0.3090059459209442
Epoch 25: Training Cost: 0.26970869302749634 CV Cost: 0.2696716785430908
Epoch 30: Training Cost: 0.23481318354606628 CV Cost: 0.234778493642807
Epoch 35: Training Cost: 0.20413875579833984 CV Cost: 0.20410452783107758
Epoch 40: Training Cost: 0.17737381160259247 CV Cost: 0.17734089493751526
Epoch 45: Training Cost: 0.154154971241951 CV Cost: 0.15412163734436035
Epoch 50: Training Cost: 0.13409043848514557 CV Cost: 0.13405777513980865
Epoch 55: Training Cost: 0.11679684370756149 CV Cost: 0.1167646050453186
Epoch 60: Training Cost: 0.10190767794847488 CV Cost: 0.10187572985887527
Epoch 65: Training Cost: 0.08908921480178833 CV Cost: 0.08905796706676483
Epoch 70: Training Cost: 0.07804617285728455 CV Cost: 0.07801502197980881
Epoch 75: Training Cost: 0.06851878017187119 CV Cost: 0.06848792731761932
Epoch 80: Training Cost: 0.06028392165899277 CV Cost: 0.06025341525673866
Epoch 85: Training Cost: 0.05315161123871803 CV Cost: 0.05312151461839676
Epoch 90: Training Cost: 0.04695956036448479 CV Cost: 0.04692988097667694
Epoch 95: Training Cost: 0.04157126694917679 CV Cost: 0.04154183343052864
Epoch 100: Training Cost: 0.03687087073922157 CV Cost: 0.03684176132082939
Epoch 105: Training Cost: 0.032760996371507645 CV Cost: 0.032732248306274414
Epoch 110: Training Cost: 0.029159190133213997 CV Cost: 0.029130620881915092
Epoch 115: Training Cost: 0.025995483621954918 CV Cost: 0.025967100635170937
Epoch 120: Training Cost: 0.023210499435663223 CV Cost: 0.023182520642876625
Epoch 125: Training Cost: 0.020754223689436913 CV Cost: 0.020726539194583893
Epoch 130: Training Cost: 0.018583718687295914 CV Cost: 0.01855628192424774
Epoch 135: Training Cost: 0.016662299633026123 CV Cost: 0.016635125502943993
Epoch 140: Training Cost: 0.0149585772305727 CV Cost: 0.014931616373360157
Epoch 145: Training Cost: 0.013445544987916946 CV Cost: 0.01341879740357399
==========
==========
Final Training cost: 0.012356716208159924
Final CV cost : 0.012330109253525734
Final Testing cost : 0.01231437548995018

12.7 Model ID: 7

Layer 1 nodes: 15
Layer 2 nodes: 150
Layer 3 nodes: 15

==========
==========

Epoch 0: Training Cost: 0.4103793799877167 CV Cost: 0.41032201051712036
Epoch 5: Training Cost: 0.38769763708114624 CV Cost: 0.387638121843338
Epoch 10: Training Cost: 0.3657759130001068 CV Cost: 0.36571818590164185
Epoch 15: Training Cost: 0.34466731548309326 CV Cost: 0.34461313486099243
Epoch 20: Training Cost: 0.3244064152240753 CV Cost: 0.3243517279624939
Epoch 25: Training Cost: 0.3050146996974945 CV Cost: 0.30496156215667725
Epoch 30: Training Cost: 0.2865019738674164 CV Cost: 0.2864479720592499
Epoch 35: Training Cost: 0.2688658535480499 CV Cost: 0.2688138782978058
Epoch 40: Training Cost: 0.2520999610424042 CV Cost: 0.2520482540130615
Epoch 45: Training Cost: 0.2361854463815689 CV Cost: 0.23613432049751282
Epoch 50: Training Cost: 0.2211022973060608 CV Cost: 0.2210523933172226
Epoch 55: Training Cost: 0.20682524144649506 CV Cost: 0.20677673816680908
Epoch 60: Training Cost: 0.1933279186487198 CV Cost: 0.1932803839445114
Epoch 65: Training Cost: 0.18058037757873535 CV Cost: 0.1805335432291031
Epoch 70: Training Cost: 0.16855303943157196 CV Cost: 0.16850726306438446
Epoch 75: Training Cost: 0.1572173833847046 CV Cost: 0.15717121958732605
Epoch 80: Training Cost: 0.14654110372066498 CV Cost: 0.1464957743883133
Epoch 85: Training Cost: 0.13649506866931915 CV Cost: 0.13645130395889282
Epoch 90: Training Cost: 0.1270524263381958 CV Cost: 0.12700866162776947
Epoch 95: Training Cost: 0.11818180978298187 CV Cost: 0.11813923716545105
Epoch 100: Training Cost: 0.10985692590475082 CV Cost: 0.10981500893831253
Epoch 105: Training Cost: 0.10205078125 CV Cost: 0.10200909525156021
Epoch 110: Training Cost: 0.09473578631877899 CV Cost: 0.0946950912475586
Epoch 115: Training Cost: 0.08788762241601944 CV Cost: 0.08784770965576172
Epoch 120: Training Cost: 0.0814811959862709 CV Cost: 0.08144202083349228
Epoch 125: Training Cost: 0.07549310475587845 CV Cost: 0.07545441389083862
Epoch 130: Training Cost: 0.06990018486976624 CV Cost: 0.06986217200756073
Epoch 135: Training Cost: 0.06468082219362259 CV Cost: 0.06464291363954544
Epoch 140: Training Cost: 0.059812869876623154 CV Cost: 0.05977614223957062
Epoch 145: Training Cost: 0.05527740344405174 CV Cost: 0.055241141468286514

==========
==========

Final Training cost: 0.05187492445111275
Final CV cost : 0.05183893069624901
Final Testing cost : 0.051811106503009796

12.8 Model ID: 8

Layer 1 nodes: 5
Layer 2 nodes: 150
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 1.6913275718688965 CV Cost: 1.6912808418273926
Epoch 5: Training Cost: 1.6376705169677734 CV Cost: 1.6375802755355835
Epoch 10: Training Cost: 1.584850549697876 CV Cost: 1.5848146677017212
Epoch 15: Training Cost: 1.533050537109375 CV Cost: 1.5330041646957397
Epoch 20: Training Cost: 1.4823286533355713 CV Cost: 1.482251763343811
Epoch 25: Training Cost: 1.432645320892334 CV Cost: 1.4326026439666748
Epoch 30: Training Cost: 1.3841238021850586 CV Cost: 1.3840736150741577
Epoch 35: Training Cost: 1.3367624282836914 CV Cost: 1.336717128753662
Epoch 40: Training Cost: 1.2905757427215576 CV Cost: 1.290529489517212
Epoch 45: Training Cost: 1.2455484867095947 CV Cost: 1.2455267906188965
Epoch 50: Training Cost: 1.2017920017242432 CV Cost: 1.201723575592041
Epoch 55: Training Cost: 1.1591421365737915 CV Cost: 1.1591078042984009
Epoch 60: Training Cost: 1.1177294254302979 CV Cost: 1.1176788806915283
Epoch 65: Training Cost: 1.077461838722229 CV Cost: 1.077408790588379
Epoch 70: Training Cost: 1.0383570194244385 CV Cost: 1.0383062362670898
Epoch 75: Training Cost: 1.0004056692123413 CV Cost: 1.0003515481948853
Epoch 80: Training Cost: 0.963568389415741 CV Cost: 0.9635246992111206
Epoch 85: Training Cost: 0.9278359413146973 CV Cost: 0.9277974367141724
Epoch 90: Training Cost: 0.8932175636291504 CV Cost: 0.8931748270988464
Epoch 95: Training Cost: 0.8596721291542053 CV Cost: 0.8596306443214417
Epoch 100: Training Cost: 0.827174186706543 CV Cost: 0.8271344900131226
Epoch 105: Training Cost: 0.7957170009613037 CV Cost: 0.7956792116165161
Epoch 110: Training Cost: 0.7652662992477417 CV Cost: 0.7652376890182495
Epoch 115: Training Cost: 0.7358276844024658 CV Cost: 0.7357898354530334
Epoch 120: Training Cost: 0.7073600888252258 CV Cost: 0.7073192000389099
Epoch 125: Training Cost: 0.6798287630081177 CV Cost: 0.6797903180122375
Epoch 130: Training Cost: 0.6532375812530518 CV Cost: 0.6531991362571716
Epoch 135: Training Cost: 0.6275332570075989 CV Cost: 0.6275070905685425
Epoch 140: Training Cost: 0.6027421355247498 CV Cost: 0.6027050018310547
Epoch 145: Training Cost: 0.5787973403930664 CV Cost: 0.5787617564201355

==========
==========

Final Training cost: 0.5602549910545349
Final CV cost : 0.5602156519889832
Final Testing cost : 0.560133159160614

12.9 Model ID: 9

Layer 1 nodes: 20
Layer 2 nodes: 100
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 1.292134165763855 CV Cost: 1.2920128107070923
Epoch 5: Training Cost: 1.263462781906128 CV Cost: 1.2633403539657593
Epoch 10: Training Cost: 1.235158085823059 CV Cost: 1.2350189685821533
Epoch 15: Training Cost: 1.207187533378601 CV Cost: 1.2070682048797607
Epoch 20: Training Cost: 1.1796518564224243 CV Cost: 1.1795201301574707
Epoch 25: Training Cost: 1.15250563621521 CV Cost: 1.1523873805999756
Epoch 30: Training Cost: 1.1258177757263184 CV Cost: 1.1256849765777588
Epoch 35: Training Cost: 1.0995283126831055 CV Cost: 1.0994195938110352
Epoch 40: Training Cost: 1.0737297534942627 CV Cost: 1.0736024379730225
Epoch 45: Training Cost: 1.048352599143982 CV Cost: 1.0482323169708252
Epoch 50: Training Cost: 1.0234205722808838 CV Cost: 1.0233163833618164
Epoch 55: Training Cost: 0.9989616870880127 CV Cost: 0.99885094165802
Epoch 60: Training Cost: 0.9749475121498108 CV Cost: 0.9748354554176331
Epoch 65: Training Cost: 0.9513859748840332 CV Cost: 0.9512653350830078
Epoch 70: Training Cost: 0.9282577037811279 CV Cost: 0.9281434416770935
Epoch 75: Training Cost: 0.9055689573287964 CV Cost: 0.9054615497589111
Epoch 80: Training Cost: 0.8833220601081848 CV Cost: 0.8832191228866577
Epoch 85: Training Cost: 0.8615085482597351 CV Cost: 0.8614086508750916
Epoch 90: Training Cost: 0.8401393890380859 CV Cost: 0.8400248885154724
Epoch 95: Training Cost: 0.8191781044006348 CV Cost: 0.8190696239471436
Epoch 100: Training Cost: 0.7986290454864502 CV Cost: 0.7985316514968872
Epoch 105: Training Cost: 0.7785194516181946 CV Cost: 0.7784113883972168
Epoch 110: Training Cost: 0.7588012218475342 CV Cost: 0.7586979269981384
Epoch 115: Training Cost: 0.7394949793815613 CV Cost: 0.7393929362297058
Epoch 120: Training Cost: 0.7205851078033447 CV Cost: 0.7204877734184265
Epoch 125: Training Cost: 0.7020772695541382 CV Cost: 0.7019758820533752
Epoch 130: Training Cost: 0.6839539408683777 CV Cost: 0.6838583946228027
Epoch 135: Training Cost: 0.666216254234314 CV Cost: 0.6661219000816345
Epoch 140: Training Cost: 0.6488624215126038 CV Cost: 0.6487644910812378
Epoch 145: Training Cost: 0.6318774819374084 CV Cost: 0.6317834854125977

==========
==========

Final Training cost: 0.618558406829834
Final CV cost : 0.6184659004211426
Final Testing cost : 0.6183744072914124

12.10 Model ID: 10

Layer 1 nodes: 20
Layer 2 nodes: 200
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 3.601768970489502 CV Cost: 3.6017582416534424
Epoch 5: Training Cost: 3.505401611328125 CV Cost: 3.505356550216675
Epoch 10: Training Cost: 3.410033941268921 CV Cost: 3.4099934101104736
Epoch 15: Training Cost: 3.315800189971924 CV Cost: 3.3157691955566406
Epoch 20: Training Cost: 3.2228407859802246 CV Cost: 3.222806692123413
Epoch 25: Training Cost: 3.1312413215637207 CV Cost: 3.1311867237091064
Epoch 30: Training Cost: 3.041079521179199 CV Cost: 3.0410006046295166
Epoch 35: Training Cost: 2.9524219036102295 CV Cost: 2.9523487091064453
Epoch 40: Training Cost: 2.865299701690674 CV Cost: 2.865251302719116
Epoch 45: Training Cost: 2.779858350753784 CV Cost: 2.779794454574585
Epoch 50: Training Cost: 2.6961090564727783 CV Cost: 2.695990800857544
Epoch 55: Training Cost: 2.614004135131836 CV Cost: 2.61391019821167
Epoch 60: Training Cost: 2.533658266067505 CV Cost: 2.5335638523101807
Epoch 65: Training Cost: 2.4550745487213135 CV Cost: 2.4549636840820312
Epoch 70: Training Cost: 2.3782403469085693 CV Cost: 2.378141164779663
Epoch 75: Training Cost: 2.3031928539276123 CV Cost: 2.3031156063079834
Epoch 80: Training Cost: 2.2300007343292236 CV Cost: 2.2298858165740967
Epoch 85: Training Cost: 2.1585307121276855 CV Cost: 2.1584455966949463
Epoch 90: Training Cost: 2.088928461074829 CV Cost: 2.088813304901123
Epoch 95: Training Cost: 2.021090030670166 CV Cost: 2.0209803581237793
Epoch 100: Training Cost: 1.9550423622131348 CV Cost: 1.9549341201782227
Epoch 105: Training Cost: 1.8907711505889893 CV Cost: 1.8906737565994263
Epoch 110: Training Cost: 1.828292727470398 CV Cost: 1.828181505203247
Epoch 115: Training Cost: 1.7675590515136719 CV Cost: 1.7674438953399658
Epoch 120: Training Cost: 1.7085657119750977 CV Cost: 1.7084410190582275
Epoch 125: Training Cost: 1.6512750387191772 CV Cost: 1.6511564254760742
Epoch 130: Training Cost: 1.5956854820251465 CV Cost: 1.595564603805542
Epoch 135: Training Cost: 1.5417648553848267 CV Cost: 1.5416412353515625
Epoch 140: Training Cost: 1.4894834756851196 CV Cost: 1.4893577098846436
Epoch 145: Training Cost: 1.4388182163238525 CV Cost: 1.4386906623840332

==========
==========

Final Training cost: 1.3994319438934326
Final CV cost : 1.399301528930664
Final Testing cost : 1.3991681337356567

12.11 Model ID: 11

Layer 1 nodes: 20
Layer 2 nodes: 15
Layer 3 nodes: 10

==========
==========

Epoch 0: Training Cost: 7.8815999031066895 CV Cost: 7.881312847137451
Epoch 5: Training Cost: 7.855218410491943 CV Cost: 7.855062961578369
Epoch 10: Training Cost: 7.829148769378662 CV Cost: 7.828918933868408
Epoch 15: Training Cost: 7.803062915802002 CV Cost: 7.8028388023376465
Epoch 20: Training Cost: 7.776969909667969 CV Cost: 7.776780128479004
Epoch 25: Training Cost: 7.751072883605957 CV Cost: 7.750790119171143
Epoch 30: Training Cost: 7.725008010864258 CV Cost: 7.724871635437012
Epoch 35: Training Cost: 7.6992506980896 CV Cost: 7.698993682861328
Epoch 40: Training Cost: 7.673461437225342 CV Cost: 7.67319917678833
Epoch 45: Training Cost: 7.647675037384033 CV Cost: 7.647456645965576
Epoch 50: Training Cost: 7.622093200683594 CV Cost: 7.621799468994141
Epoch 55: Training Cost: 7.596343040466309 CV Cost: 7.596198081970215
Epoch 60: Training Cost: 7.570882797241211 CV Cost: 7.570636749267578
Epoch 65: Training Cost: 7.5454325675964355 CV Cost: 7.545149326324463
Epoch 70: Training Cost: 7.519964694976807 CV Cost: 7.51976203918457
Epoch 75: Training Cost: 7.494694709777832 CV Cost: 7.494394779205322
Epoch 80: Training Cost: 7.469296932220459 CV Cost: 7.469101905822754
Epoch 85: Training Cost: 7.444127559661865 CV Cost: 7.44389533996582
Epoch 90: Training Cost: 7.418989658355713 CV Cost: 7.4187211990356445
Epoch 95: Training Cost: 7.393829345703125 CV Cost: 7.393612861633301
Epoch 100: Training Cost: 7.36889123916626 CV Cost: 7.368592262268066
Epoch 105: Training Cost: 7.343788146972656 CV Cost: 7.343604564666748
Epoch 110: Training Cost: 7.318909645080566 CV Cost: 7.31868839263916
Epoch 115: Training Cost: 7.294127464294434 CV Cost: 7.293829441070557
Epoch 120: Training Cost: 7.269244194030762 CV Cost: 7.269026279449463
Epoch 125: Training Cost: 7.244594573974609 CV Cost: 7.244307994842529
Epoch 130: Training Cost: 7.219847679138184 CV Cost: 7.219613075256348
Epoch 135: Training Cost: 7.195223808288574 CV Cost: 7.195019721984863
Epoch 140: Training Cost: 7.1707444190979 CV Cost: 7.170435428619385
Epoch 145: Training Cost: 7.146156311035156 CV Cost: 7.145944118499756

==========
==========

Final Training cost: 7.126615524291992
Final CV cost : 7.126371383666992
Final Testing cost : 7.1258368492126465