STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

# Application of Deep Learning in Credit Fixed Income Markets

*Final Presentation*

Ivan Bakrac
FA 800, Project in Financial Analytics, Fall'20
Advisor: Dr.Khaldoun Khashanah

# Table of Contents



# Project Overview

# Project Overview

- Credit markets market data flows are **inefficient** in many respects to enable **robust coverage and precision for AI bond pricing.**
- Credit Markets are reliant on segregated and manual data operations between counterparties creating **disparate data sets.**
- **Goal:** real time **predictive analytics using deep learning** focused on pricing corporate credit given security characteristics, features and market dynamics.

Credit Trading Process:

1. **Electronic RFQ - Through a Bloomberg (or other) terminal window**
   a) Electronic RFQ windows appear on top of each other as per the time when the RFQ was initiated
   b) The execution trader can confirm and submit the price suggested by the RFQ response, based on his/her knowledge of the bond market gathered during the day
   c) The execution trader may also check a third-party pricing source to validate the pricing on the RFQ responses

2. **Manual RFQ - Through a phone call**
   a) The security is checked on the Bloomberg terminal (or other) with the ISIN, as well as the "Quotes" on the terminal from other dealers, the best price to be accepted.
   b) The phone call is treated as a priority and the electronically received RFQ responses are put on hold until a quote has been recorded.

# Literature Overview

# Literature Overview

- Machine learning methods **improve out of sample** returns predictions for **fixed income corporate bonds** excess returns.[3]
- Deep learning is used for **prepayment modeling** for mortgages with high degree of success relative to traditional methods.[1]
- Predictive gains are traced to allowance of **complex nonlinear predictor interactions**.[1]
- **Nonlinear machine learning methods** are suitable for **credit market** due to more sensitivity to **nonlinear payoff** and downside risk[3]
- Overall, deep learning is able to **model nonlinear relationships** and highly **interactive asset risk factors**[2]

# Data Preprocessing

# Bond Ticker x Input Features (2D Tensor)

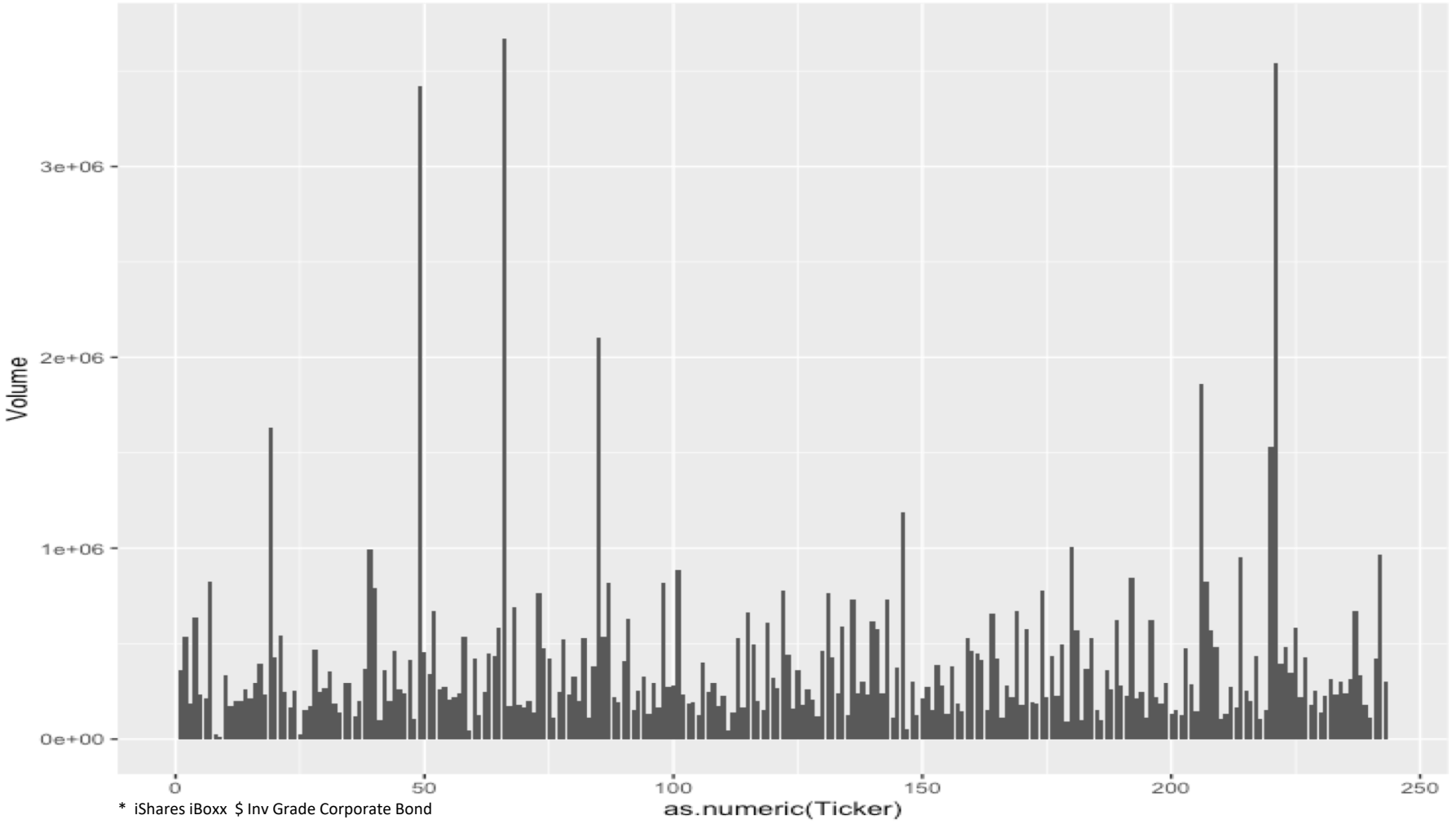
Credit fundamentals | Security Description | Spread, Govt Curve, Trace Price

| | tracematch | tot_debt_to_tot_asset | net_debt_to_ebitda2 | rtg_sp | ticker | industry_sector | mty_years | cpn | yas_mod_dur | oas_spread | yield | price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | BBG00000HRQ2 | 31.72 | 6.25 | 13 | 70 | 1 | 28.6 | 4.8 | 16.2 | 164.288225 | 2.207830 | 141.865 |
| 1 | BBG00000PQ76 | 54.21 | 4.46 | 13 | 163 | 2 | 21.9 | 4.5 | 14.0 | 233.487551 | 3.086082 | 155.156 |
| 2 | BBG00000R0K6 | 40.01 | 3.77 | 11 | 8 | 4 | 4.1 | 3.5 | 3.5 | 197.793935 | 2.876434 | 146.848 |
| 3 | BBG000013MV0 | 35.35 | 7.04 | 16 | 95 | 6 | 11.6 | 4.6 | 8.8 | 257.803027 | 3.294753 | 132.493 |
| 4 | BBG00001SH53 | 2.63 | 12.50 | 3 | 137 | 6 | 21.8 | 4.1 | 15.0 | 108.192348 | 1.960752 | 147.380 |
| 5 | BBG00001W439 | 72.95 | 1.34 | 1 | 106 | 3 | 6.5 | 2.5 | 5.9 | 115.412631 | 2.132883 | 150.889 |
| 6 | BBG00001WGX9 | 42.41 | 3.46 | 16 | 147 | 6 | 5.5 | 2.2 | 4.3 | 122.279237 | 1.882232 | 154.815 |
| 7 | BBG00003WCV6 | 45.43 | 6.76 | 11 | 123 | 5 | 10.3 | 2.0 | 9.2 | 277.375091 | 3.461006 | 139.486 |
| 8 | BBG00005JCP3 | 16.43 | -0.87 | 7 | 58 | 2 | 3.4 | 3.6 | 3.2 | 129.689945 | 2.349620 | 152.480 |
| 9 | BBG00005SG74 | 22.01 | 2.14 | 13 | 12 | 6 | 4.5 | 5.8 | 3.9 | 270.884862 | 3.415596 | 141.674 |
| 10 | BBG000065XF1 | 35.35 | 7.04 | 16 | 95 | 6 | 4.6 | 3.5 | 4.1 | 303.334338 | 4.050932 | 136.170 |
| 11 | BBG000067ZF4 | 31.72 | 6.25 | 13 | 70 | 1 | 30.1 | 3.6 | 18.1 | 196.881381 | 2.487460 | 141.641 |
| 12 | BBG00006BTL6 | 46.31 | 3.46 | 13 | 101 | 6 | 5.0 | 4.3 | 4.5 | 116.540278 | 1.933033 | 145.581 |
| 13 | BBG00006KZ41 | 22.45 | 0.83 | 1 | 53 | 5 | 24.1 | 4.3 | 15.2 | 146.317393 | 2.050691 | 137.763 |
| 14 | BBG00006RMZ9 | 29.96 | 2.10 | 16 | 42 | 6 | 4.5 | 3.3 | 4.2 | 164.207678 | 2.694131 | 179.451 |
| 15 | BBG00006SFH3 | 50.28 | 6.26 | 14 | 90 | 9 | 29.3 | 3.4 | 18.1 | 282.594832 | 3.507682 | 135.086 |
| 16 | BBG00007MG21 | 29.96 | 2.10 | 16 | 42 | 6 | 10.3 | 2.7 | 8.2 | 172.903679 | 2.438139 | 142.079 |
| 17 | BBG00007YQN3 | 13.36 | -6.49 | 6 | 37 | 6 | 30.0 | 2.9 | 20.0 | 108.712185 | 2.161621 | 156.101 |
| 18 | BBG000091HK8 | 2.63 | 12.50 | 3 | 137 | 6 | 3.5 | 3.6 | 3.3 | 109.101482 | 1.908977 | 153.374 |
| 19 | BBG000092845 | 45.43 | 6.76 | 11 | 123 | 5 | 27.4 | 5.2 | 15.3 | 333.813833 | 4.302380 | 132.018 |

Source : Bloomberg and Trace

# Volume Snapshot of LQD* Tickers

Trace Execution Volume (intraday sample per issuer)




* iShares iBoxx $ Inv Grade Corporate Bond

Source : Trace

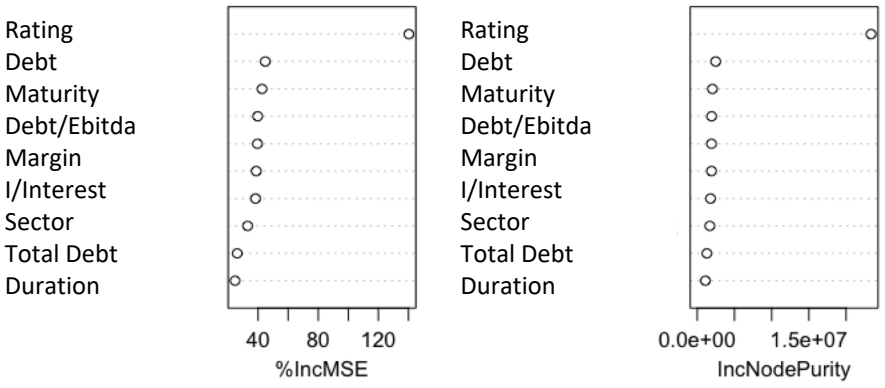
# Methods: Benchmark & Deep Learning Models

# Benchmark Models: Random Forest & Linear

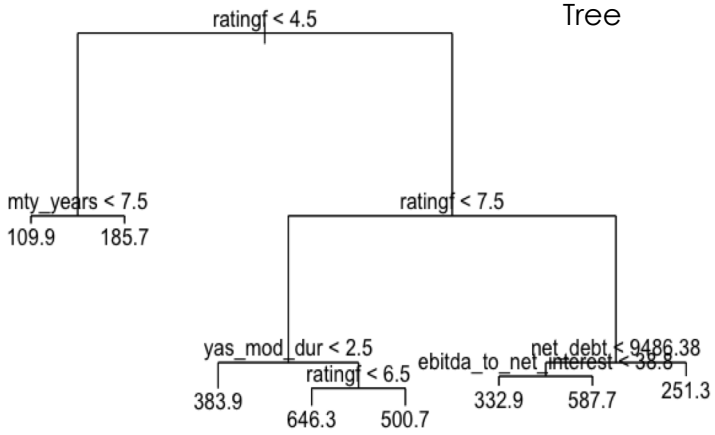
- Moving from linear to nonlinear model **improves MSE (mean squared error)**
- **Random Forest** yields best results with lowest MSE.

| Model | Test MSE |
|---|---|
| Linear Model Regression[1] | 28,897 |
| Decision Tree Regression | 14,915 |
| **Random Forest** | **5,968** |

Feature Importance



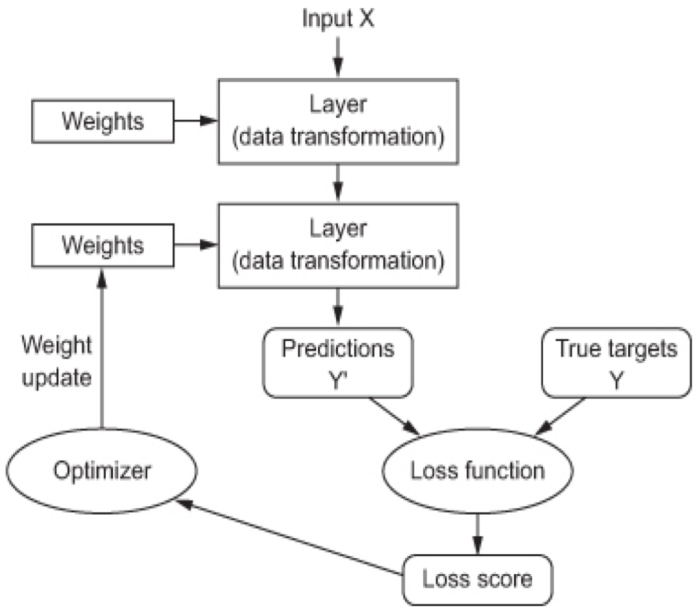

Tree




[1] Appendix, Target variable = Price

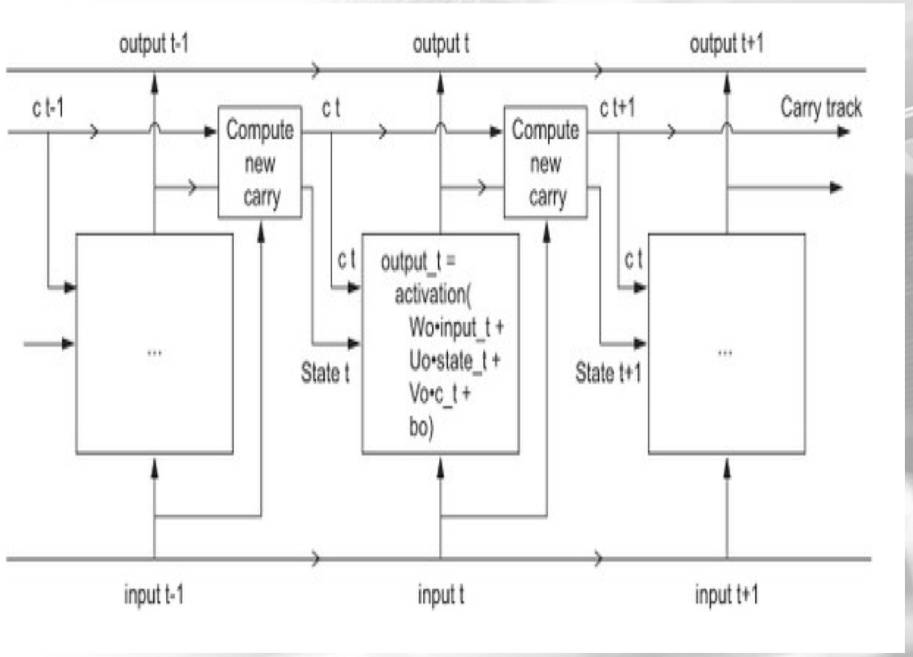
# Deep Learning Methods - Anatomy

## Multilayer Perception (MLP)




Reference [4]

## Long Short Term Memory (LSTM)

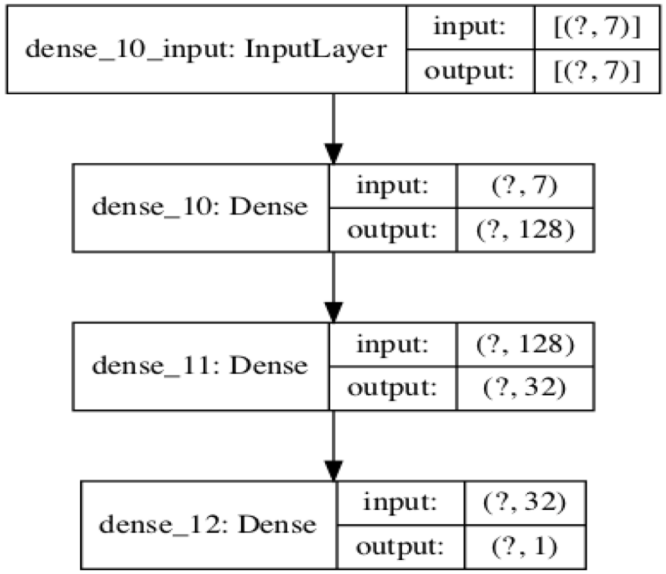


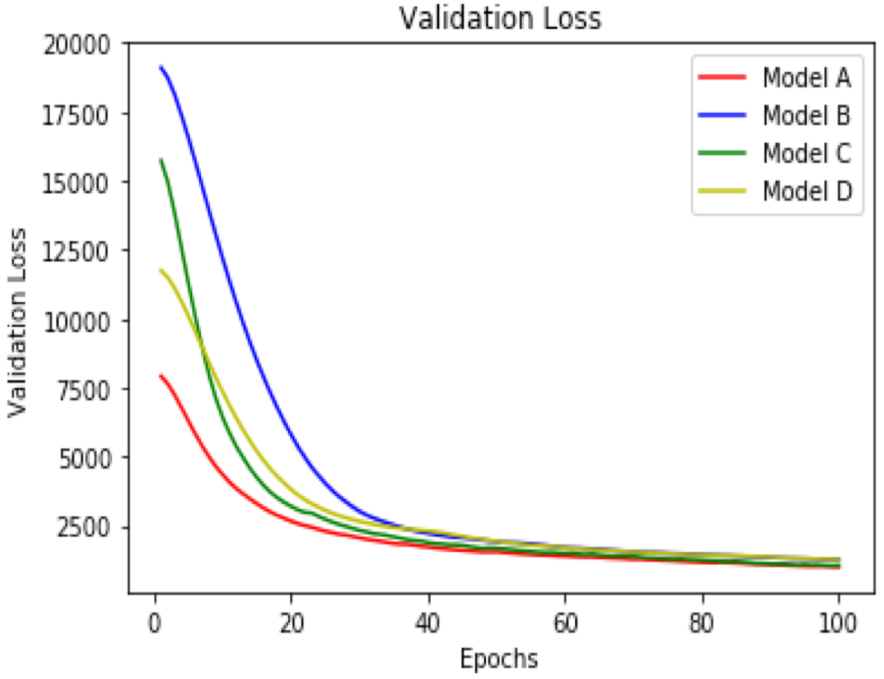

Reference [4]

# Deep Learning I: Multilayer Perception (MLP)

| Model | MSE Test | MAE[1] Test | Architecture: Layers, Nodes, Activation, O.Activation, L1/2 |
|---|---|---|---|
| Model A | 1179 | 28 | 1,256, Rectified Linear Unit (RELU) ,Regression (None) |
| Model B | 1059 | 25 | 1,128, Rectified Linear Unit (RELU) ,Regression (None), Dropout (0.5) |
| Model C | 1158 | 28 | 2,128,32, Rectified Linear Unit (RELU), Regression (None), L2 (L=0.001) |
| Model D | 896 | 18 | 1,128, Rectified Linear Unit (RELU) ,Regression (None) |

Keras Model Architecture (Model C)



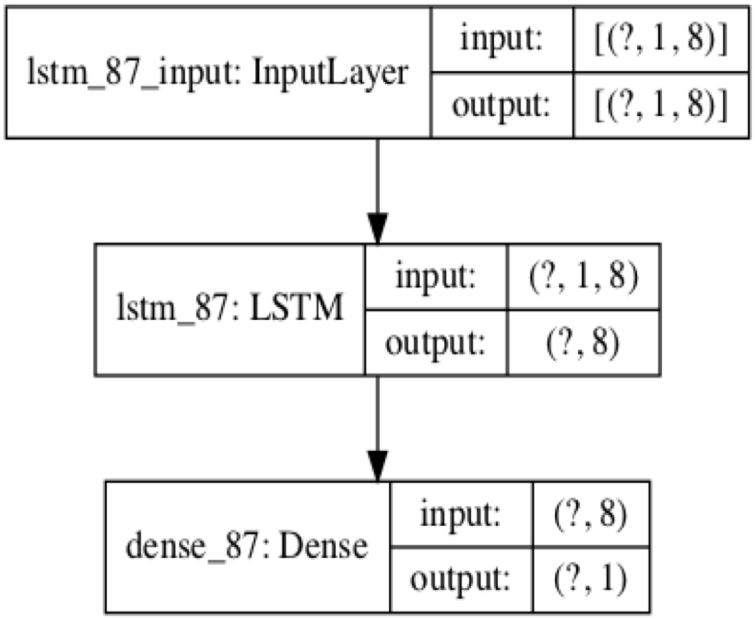



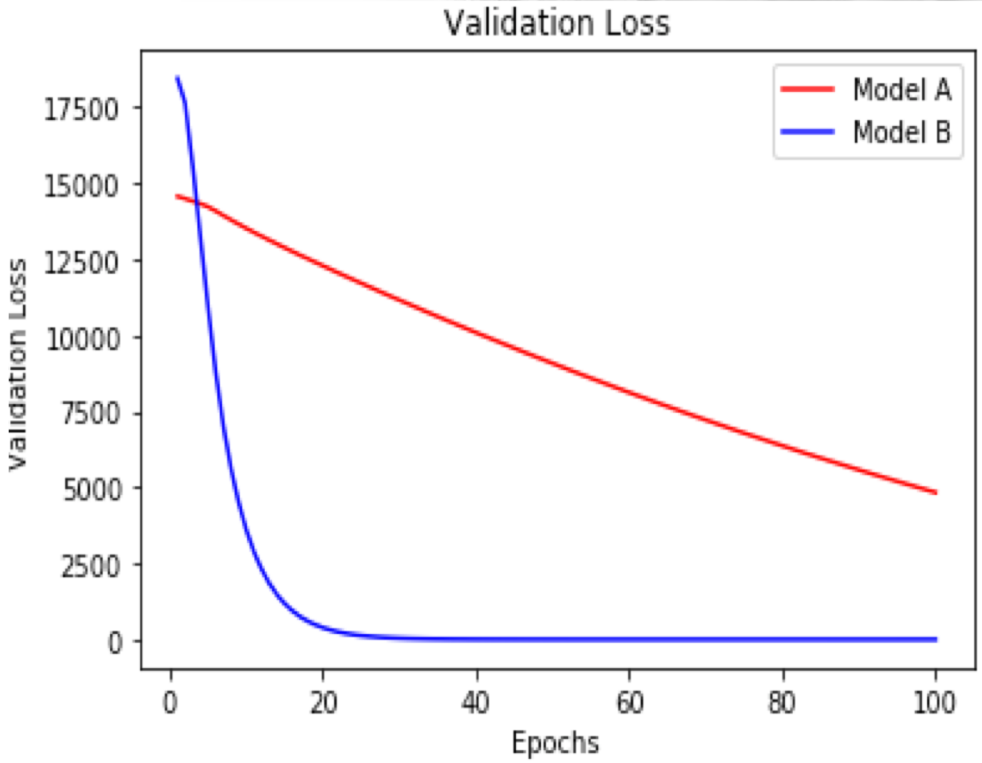

[1] MAE= Mean Absolute Error , Loss Function = Mean Squared Error

# Deep Learning II: Long Short Term Memory (LSTM)

| Model | MSE Test | MAE[1] Test | Architecture: Layers, Timesteps, Features, Activation, L1/2 |
|---|---|---|---|
| Model A | 5300 | 25 | 2,2,8 Rectified Linear Unit (RELU) ,Regression (None), Dropout (0.5) |
| **Model B** | **7.68** | **0.7** | **1,1,8, Rectified Linear Unit (RELU) ,Regression (None), None** |

Keras Model Architecture (Model B)




[1] MAE= Mean Absolute Error , Loss Function = Mean Squared Error

# Conclusions & Challenges:

**Major Conclusions:**

- **Linear** models have relatively **high mean squared error**.
- Moving from **linear to nonlinear models** improves prediction.
- **Random forest** offers alternative approach to linear models.
- Deep learning models **outperform** linear and nonlinear (RF) models.
- Deep Learning models are able to **converge per loss functions.**
- **Extending MLP model to LSTM** deals with **"temporal" relationships.**

**Major Challenges:**

- **Disparate data sources** remain an issue.
- Infrequent Trades **(data coverage issue).**
- While loss curve is **converging**, overall **MSE still relatively high.**
- Stacking **nonstationary** data with some **static** features.

# References & Appendix

# References


- [1] Jiawei Zhang, Xiao, Jian Zhao, Joy Zhang, Fei Teng, Siyu Lin, and Hongyuan Li. Agency MBS Prepayment Model Using Neural Networks. *The Journal of Structured Finance*, 24(4):17–33, 2019.
- [2] Shihao Gu, Bryan Kelly, Dacheng Xiu. Empirical Asset Pricing via Machine Learning . *The Review of Financial Studies*, 33(5):1533–1575, 2020.
- [3] Turan G. Bali, Amit Goyal, Dashan Huang, Fuwei Jiang, Quan Wen. The Cross-Sectional Pricing of Corporate Bonds Using Big Data and Machine Learning . *Georgetown University, Central University of Finance and Economics (CUFE), Singapore Management University,*2020.
- [4] Chollet François. *Deep Learning with Python*. Manning Publications, 2018
- [5] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press, 2016

# Appendix : General Modelling Steps

**Step 1:** Implement  linear model with best subset selection on features.

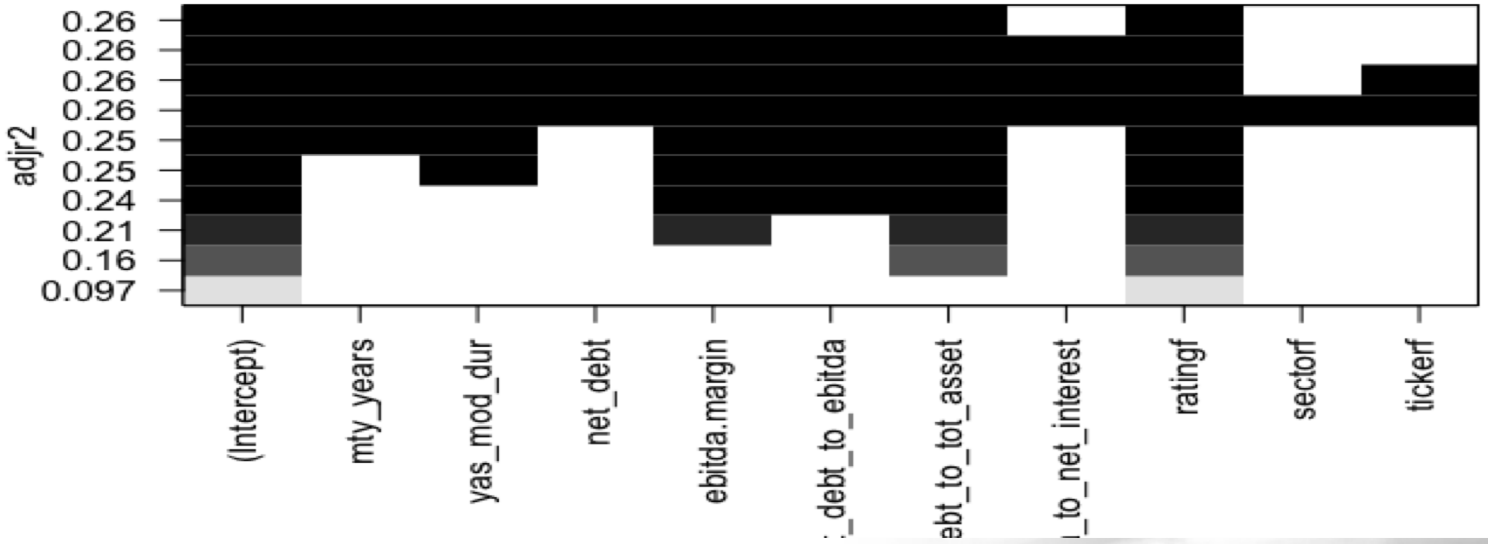
**Step 2**: Extend linear model  to nonlinear model – Regression Tree

**Step 3**: Extend Regression Tree to Random Forest (bagging)

**Step 4:** Implement Deep Learning MLP Model (Feed Forward Net)

1. Data is split into train, test, validation set
2. Hyperparameters  tuning with validation set

# Appendix : Linear Model Subset Selection



# Appendix : MLP Model Architecture

```
# ================================================================================================================
modela1=models.Sequential()
modela1.add(layers.Dense(256, activation='relu', input_shape=(11,)))
modela1.add(layers.Dense(1))
modela1.summary()
# ================================================================================================================
modelb= models.Sequential()
modelb.add(layers.Dropout(0.00001, input_shape=(11,)))
modelb.add(layers.Dense(128, activation='relu'))
modelb.add(layers.Dropout(rate = 0.5))
modelb.add(layers.Dense(1))
modelb.summary()
# ================================================================================================================
modelc=models.Sequential()
modelc.add(layers.Dense(128, activation='relu', input_shape=(11,)))
modelc.add(layers.Dense(32, activation='relu'))
modelc.add(layers.Dense(1))
modelc.summary()
# ================================================================================================================
modelb2= models.Sequential()
modelb2.add(layers.Dense(128, activation='relu',kernel_regularizer=regularizers.l2(0.001),input_shape=(11,)))
modelb2.add(layers.Dense(1))
modelb2.summary()
```

# Appendix : LSTM Model Architecture

```
# ================================================================================================================
modelLSTM1 = models.Sequential()
modelLSTM1.add(layers.LSTM(9, activation='relu',return_sequences=True,input_shape=(9,1),dropout=0.5))
modelLSTM1.add(layers.LSTM(9))
modelLSTM1.add(layers.Dense(1))
modelLSTM1.summary()
# ================================================================================================================
modelLSTM2 = models.Sequential()
modelLSTM2.add(layers.LSTM(9, activation='relu', input_shape=(9,1)))
modelLSTM2.add(layers.Dense(1))
modelLSTM2.summary()
# ================================================================================================================
# stacked lstm
modelLSTM3 = models.Sequential()
modelLSTM3.add(layers.LSTM(100, activation='relu', return_sequences=True, input_shape=(9, 1)))
modelLSTM3.add(layers.LSTM(50, activation='relu', return_sequences=True))
modelLSTM3.add(layers.LSTM(25, activation='relu', return_sequences=True))
modelLSTM3.add(layers.LSTM(25, activation='relu'))
modelLSTM3.add(layers.Dense(20, activation='relu'))
modelLSTM3.add(layers.Dense(10, activation='relu'))
modelLSTM3.add(layers.Dense(1))
```