Stevens Institute of Technology

Financial Analytics

Application of Deep Learning in Credit Fixed Income Markets

Author:
Ivan Bakrac

Supervisor:
Dr.Khaldoun Khashanah

December 21, 2020

Abstract

In this paper I provide comprehensive study of deep learning methods application in credit fixed income markets and its potential use for real time predictive analytics. I examined how classical and deep learning machine learning methods can be used for real time pricing of investment grade corporate bonds. Firstly, using set of fixed income securities characteristics, features and market dynamics I find that nonlinear machine learning methods substantially improve out-of-sample price prediction compared to linear regression models. Secondly, I find that deep learning models outperform classical machine learning models, including linear best subset selection methods and non-linear models such as decision trees and random forest regressions.

Keywords: deep learning, multilayer perception, long short term memory, best subset selection, random forest, decision tree, investment grade corporate bonds

1 Introduction

Credit fixed income markets are less liquid relative to equity markets and currently trading on over the counter market and with exponential increase in electronic trading over the last few years. Global credit markets are diverse in terms of features and security characteristics such as coupon, maturity, sector, subordination, ratings, credit risk as well as fundamental characteristics such as leverage, total debt and interest coverage. However, credit markets trading data flows are inefficient in many respects to enable robust coverage and precision for machine learning approach to bond pricing. Bond pricing is reliant on segregated and manual data operations between counterparties creating disparate data sets with increased shift to electronic trading over the last several years. The goal of this research is to show potential use of predictive analytics using deep learning methods and assess its potential use. The focus is on understanding of pricing corporate credit given security characteristics, features and market dynamics with deep learning approach and comparing them to classical machine learning methods.

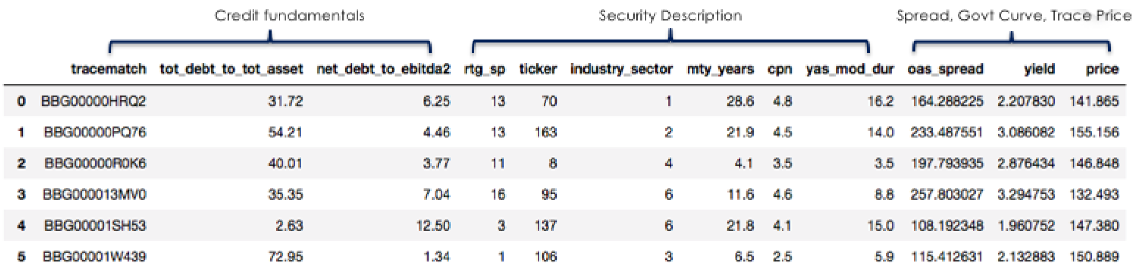
Bali et al. (2020) study has shown that machine learning methods improve out of sample returns predictions for fixed income corporate bonds excess returns. The study concluded that machine learning is suitable for credit markets due to more sensitivity to nonlinear payoff and downside risk of fixed income securities where maximum upside at maturity is par and the downside is bond default or recovery rate. This study concluded that neural nets outperformed linear models and other classical machine learning models. Another recent study done by Zhang et al. (2019) utilized deep learning for prepayment modeling in agency mortgage backed securities with high degree of success compared to industry econometric prepayment models used over the last fifteen years. The backtesting results produced more accurate prepayment speeds applied to mortgage backed securities compared to econometric methods. Gu et al. (2020) have shown comparative analysis in empirical asset pricing applications across machine learning models ranging from classical to deep learning methods applied to equities. This context can also be related to fixed income markets with respect to feature selection as the study selected range of security specific attributes for feature selection and engineering. Their conclusion is that deep learning and nonlinear models outperform classical linear machine learning models and trace predictive gains to presence of complex nonlinear predictor interactions. Overall, current literature review suggests that deep learning is able to model nonlinear relationships in asset pricing and learn highly interactive asset risk factors with clear application in fixed income markets.

2 Methodology

2.1 Data Preprocessing

Given disparate data source, a major challenge for credit markets is data collection. Data source for this study is Bloomberg for market characteristics and Trace reporting database for pricing. In order to make this study possible, I focused on liquid investment grade credit market represented by LQD (iShares iBoxx Investment Grade Corporate Bond ETF). Per Figure 1, data is comprised of fundamental metrics, security description metrics and market data. The basic idea is to combine all three categories in feature selection process as these characteristics define fixed income securities and let the deep models decide the most relevant features. Figure 2 shows volume for credit fixed income and its availability for about 250 credit tickers. Final data set blends together features and pricing where pricing is actual executed price which is target variable in machine learning models. This is critical as indicative pricing from Bloomberg source might not be actual executed price and pricing for many bonds is stale in nature as unlike in equity, turnover in credit market tends to be lower.

Figure 1: Fixed Income Features, Source: Bloomberg, Trace



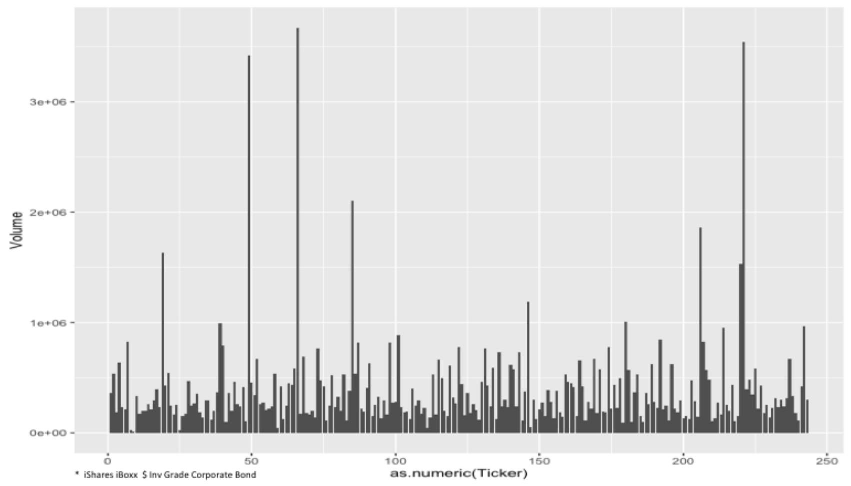
	Credit fundamentals			Security Description						Spread, Govt Curve, Trace Price		
	tracematch	tot_debt_to_tot_asset	net_debt_to_ebitda2	rtg_sp	ticker	industry_sector	mty_years	cpn	yas_mod_dur	oas_spread	yield	price
0	BBG00000HRQ2	31.72	6.25	13	70	1	28.6	4.8	16.2	164.288225	2.207830	141.865
1	BBG00000PQ76	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

2.2 Classical Machine Learning Methods

In order to assess relevance of deep learning methods where they self-learn and transform features I start the analysis with simple linear models, then extend the analysis to nonlinear models such as decision tree, extend the models to random forest decision trees and use these these models as performance benchmarks for deep learning methods. That is, I evaluate family of linear (in this case best subset selection) and nonlinear models (random forest and simple decision tree).

The first step in this process is to assess feature importance using linear model selection with best subset selection method. To perform best subset selection, I fit a separate least squares regression for each possible combination of the p predictors to see relevant features. Figure 3 shows feature selection based on this method. Adjusted

Figure 2: Trace Volume, Source: Trace Reporting Database



R squared is significant explaining cumulative R-squared of close to 30%. Figure 4 shows feature selection based on this method. Adjusted R squared is significant but peaking with around eight features.

The second step is to evaluate nonlinear group of models. For this purpose I picked decision tree along with random forest (ensemble method consisting of 1,000 trees). These nonlinear group of models do not have fully connected layers like deep learning methods. The impurity of a node is measured by the least-squared deviation. Decision tree works well when there is nonlinear relationships between variables. It does not require linearity assumption ant it is not sensitive to outliers. It assumes all independent variables interact each other which sometimes is not the case. Figure 5 shows feature importance and shows that metrics such as rating, maturity, total debt and leverage are critical components of the model self-selection selection process.

Figure 3: Best Subset Selection, Cumulative R-squared

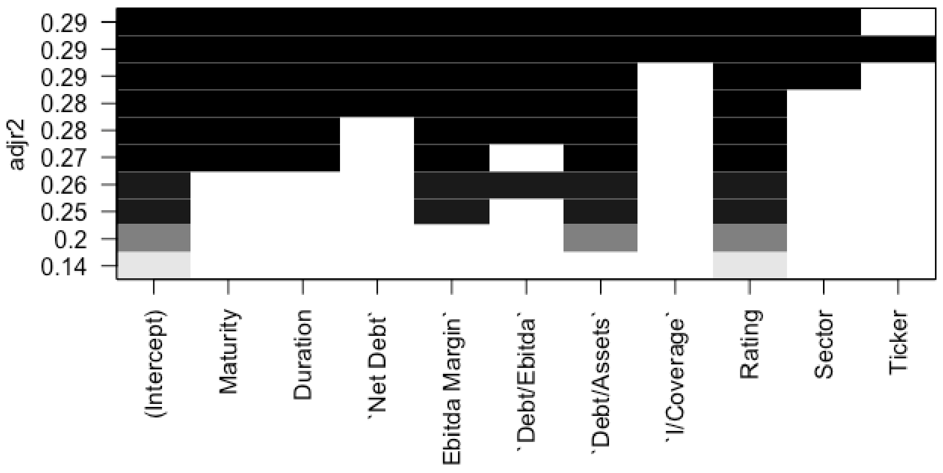


Figure 4: Feature Selection and R-squared, Index: Number of Features

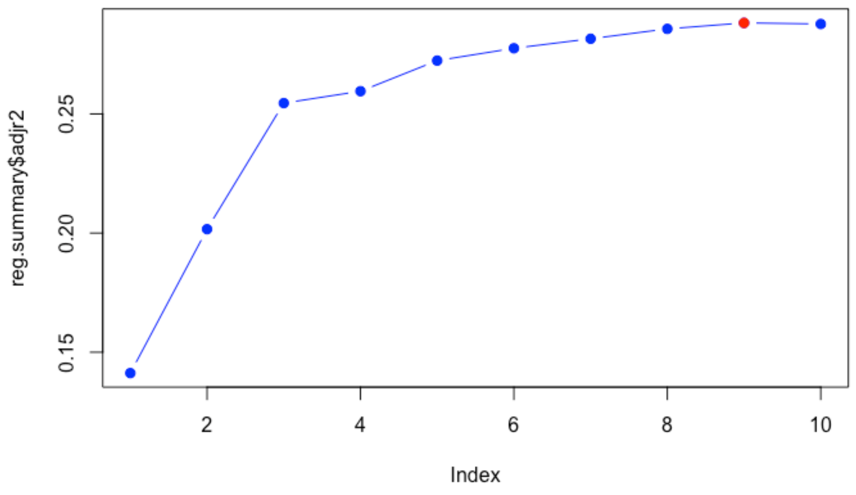
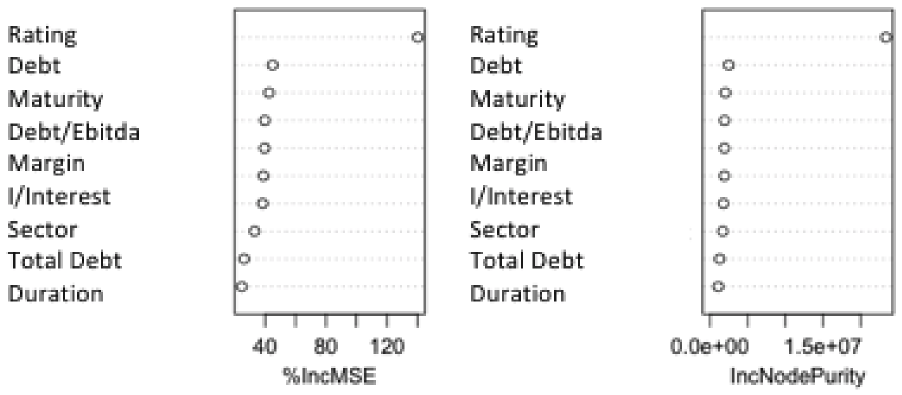


Figure 5: Random Forest Feature Selection



In summary, Table 1 shows out of sample test results. MSE, mean squared error, is relatively high and while the analysis demonstrates features importance, using these models for pricing corporate bonds yields relatively poor results from practical standpoint. Allowing for nonlinearities substantially improves predictions by expanding linear models to accommodate for nonlinear predictive relationships via the random forest regression trees.

Table 1: Test Mean Squared Error

	Method	MSE
1	Best Subset Selection	28,897
2	Decision Tree Regression	14,514
3	Random Forest Regression	5,698

2.3 Deep Learning Methods

For the deep learning models, I consider two architectures: MLP or multilayer perception, a fully connected dense architecture and LSTM or long short term memory model. Figure 6 shows anatomy of two model architectures.

MLP consists of an input layer of raw predictors, one or more hidden layers that interact and nonlinearly transform the predictors, and an output layer that aggregates hidden layers into an ultimate outcome prediction. The number of units in the input layer is equal to the dimension of the predictors, which is set to eight in this example, representing fixed income input features (market, technical and fundamental features). The activation function used is RELU (rectified linear unit). The rectified linear activation function is a simple calculation that returns the value provided as an input directly, or the value zero if the input is zero or less. Given that this is regression task, final function output function is one dense layer with activation function set as linear. Table 2 shows network architecture for MLP. I explore four different models given empirical nature of training deep leaning models.

LSTM is a sequence model that deals with "temporal relationships", something that multilayer perception model is not able to do effectively. LSTM is a special kind of recurrent neural net, capable of learning long-term dependencies with information contained in a gated cell. LSTM gates determine how long to hold onto old information, when to remember and forget, and how to make connections between old memory with the new input. Figure 7 shows architecture of two models. Eq 1 demonstrates basic idea where new state is updated by both input at time t and prior state of the network:

$$h_t = f_w(h_{t-1}, x_t) \tag{1}$$

Both models (MLP and LSTM) have the loss function as MSE (mean squared error) with RMSprop as their optimizer. RMSprop is a version of stochastic gradient descent that takes small steps along the loss surface following the gradient until it finds a minimum along with adaptive learning rate. The size of the step is called the learning rate and the larger the learning rate the faster it learns and training is performed over multiple epochs. Given D, final dense layer and S, the layer activation function (RELU), the training loss in Eq 2 is defined as:

$$\mathfrak{L} = \frac{1}{N} \sum_i D(S(WX_i + b)), L_i) \tag{2}$$

Table 2: MLP Model Architecture

Model	MSE Test	MAE	Activation	Regularization	Layers	Nodes
Model A	896	19	RELU	None	1	256
Model B	1059	25	RELU	Dropout(0.5)	1	128
Model C	1158	28	RELU	L2=0.001	2	128,32
Model D	1179	28	RELU	None	1	128

Figure 6: Model Anatomy, Reference: Deep Learning by François Chollet

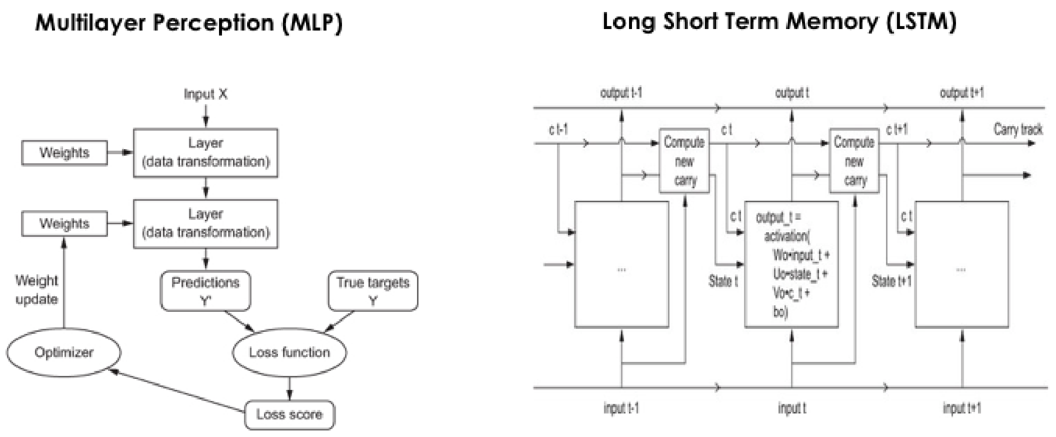


Table 3: LSTM Model Architecture

Model	MSE Test	MAE	Activation	Regularization	Layers	Steps
Model A	5300	25	RELU	Dropout (0.5)	2	2
Model B	7.68	0.7	RELU	None	1	2

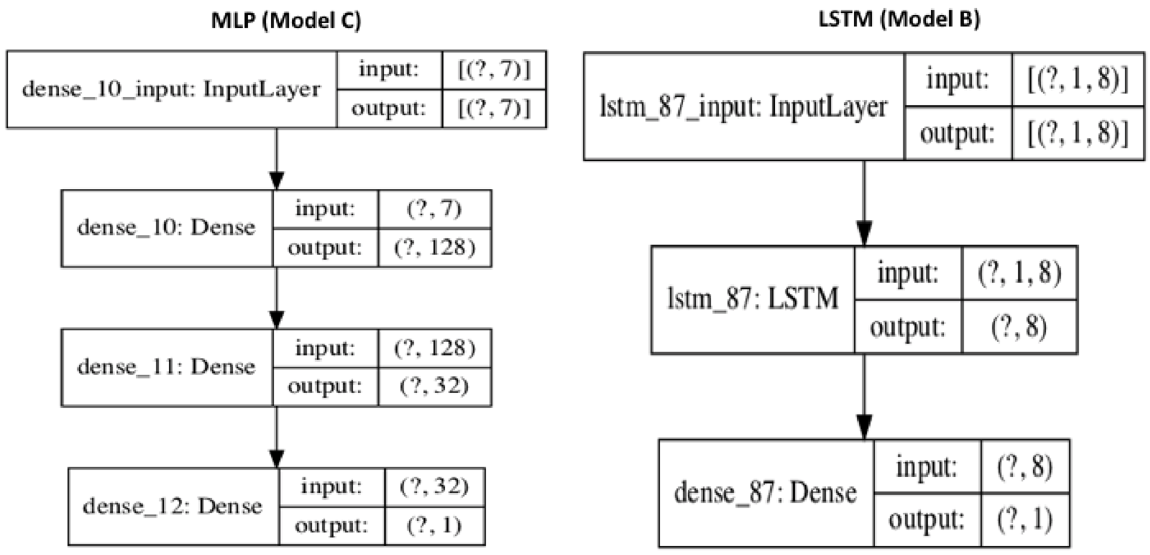
3 Results

Table 2 and 3 show results from MLP and LSTM respectively and model architecture for out of sample test. Here, MSE (mean squared error) is loss function and MAE (mean absolute error) is evaluation metric. Figure 8 shows convergence of loss curves on the validation data (the data is split into 60% training, 20% validation, and 20% testing). Both models are able to converge with LSTM outperforming MLP as MAE is 0.7 which is significantly lower than MSE for MLP models. With that being said, both models show "correct" learning curve as they are able to converge from minimizing loss function perspective. With regards to MLP, all four models deliver similar results with Model A, architecture with one hidden layer, outperforming. With regards to LSTM, structure with one hidden layer tends to work well and delivers the best results. More complicated LSTM structures underperform in this training. Overall, training shows "shallow" outperforms "deep" learning model structures with sequence models demonstrating promising results.

4 Conclusion

Linear models have relatively high mean squared error as well as high mean absolute error. Moving from linear to nonlinear models improves prediction for credit pricing. This confirms prior studies of existence of nonlinear interactions among risk factors.

Figure 7: Model Architecture (Keras)



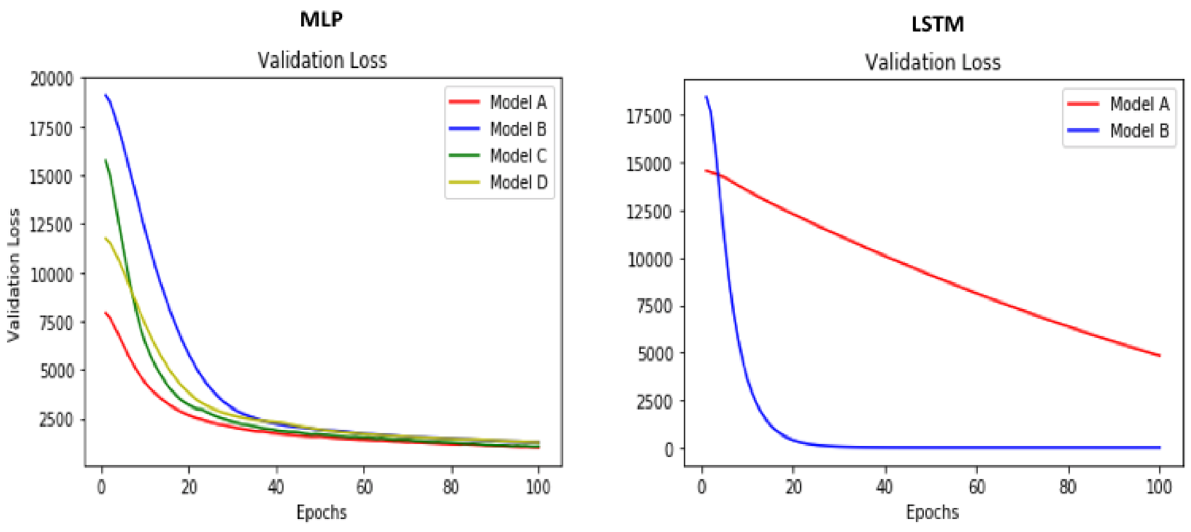
Random forest offers alternative approach to linear models and while mean absolute error is high, it is useful model from feature selection and engineering standpoint and assessing non-linearity. In addition to that, it serves as an intermediate step between linear and deep learning models. Both group of models offer perspective in terms of feature selection and engineering and model selection as one can get intuitive understanding of features interactions. Deep learning models outperform linear and nonlinear machine learning models. Deep learning models are able to converge per loss functions. Extending multilayer perception model to long short term memory deals with "temporal" relationships in the data and can significantly improve prediction with even two times steps included as in this study. Major challenges remain disparate data sources as well as infrequent trades. While loss curve is converging, overall mean absolute error is still relatively high. From that standpoint, LSTM offers superior performance relative to multilayer perception and classical machine learning methods even with limited number of time steps included in the model. This also makes intuitive sense as the goal is to incorporate recent time steps (recent executed prices) in the process reflecting changes in market dynamics. Further area of study might include broad market variables as such as credit derivatives indices, overall change in government yields and similar to reflect on most recent market changes. Even though specific corporate bonds might not trade at the same time as broad market indices, adding highly liquid market focused features may improve model performance especially with LSTM architecture that can handle sequence data.

Major code references: appendix A and B.

A R code listing

```
# Best Subset Selection Methods
```

Figure 8: Validation Loss Curves



```r
regfit.full=regsubsets(px ~ ., data=credit.train ,nvmax=14)
t(summary(regfit.full)$which)
coefficients(regfit.full ,8)
summary(regfit.full)$adjr2
reg.summary=summary(regfit.full)
which.max(reg.summary$adjr2)
plot(reg.summary$adjr2,type='b',col="blue",pch=19)
points(which.max(reg.summary$adjr2),reg.summary$adjr2[which.max(reg.
    summary$adjr2)],col="red",pch=19)
plot(regfit.full ,scale="adjr2")
names(credit)
# Decision Tree Method
library(tree)
tree.credit = tree(px~ ., data = credit.train)
summary(tree.credit )
plot(tree.credit )
text(tree.credit , pretty = 0)
set.seed(personal)
cv.credit  = cv.tree(tree.credit , FUN = prune.tree)
par(mfrow = c(1, 2))
plot(cv.credit$size, cv.credit$dev, type = "b")
plot(cv.credit$k, cv.credit$dev, type = "b")
# Best size = 9
pruned.credit_tree = prune.tree(tree.credit, best = 8)
par(mfrow = c(1, 1))
plot(pruned.credit_tree)
text(pruned.credit_tree, pretty = 0)
pred.pruned_credit = predict(pruned.credit_tree, credit.test)
mean((credit.test$px - pred.pruned_credit)^2)
# Random Forest Method
library(randomForest)
```

```r
set.seed(personal)
bag.credit=randomForest(px ~ ., data = credit.train,mtry=10,
    importance=TRUE)
bag.credit
yhat.bag = predict(bag.credit,newdata=credit.test)
plot(yhat.bag, credit.test$px)
abline(0,1)
mean((yhat.bag-credit.test$px)^2)
importance(bag.credit)
varImpPlot(bag.credit)
```

B Python code listing

```python
import random
import pandas as pd
import numpy as np
from tensorflow.random import set_seed # tf version = 1.15.0
from keras import models
from keras import layers
from keras import regularizers
random.seed(30)

# Fit Function
def fit_model(model, x_train, y_train, x_val, y_val, x_test, y_test)
    :
    model.compile(optimizer='rmsprop',loss='mse',metrics=['
    mean_absolute_error'])
    history = model.fit(x_train, y_train,
                                        epochs=100,
                                        batch_size=512,
                                        validation_data=(x_val, y_val))
    results = model.evaluate(x_test, y_test)
    print('Test loss:', results[0])
    print('Test accuracy:', results[1])

    model.predict(x_test)
    print(model.predict(x_test)[1:10])

    return history

# Model Architecture MLP
modela1=models.Sequential()
modela1.add(layers.Dense(256, activation='relu', input_shape=(7,)))
modela1.add(layers.Dense(1))
modela1.summary()

modelb= models.Sequential()
modelb.add(layers.Dropout(0.00001, input_shape=(7,)))
modelb.add(layers.Dense(128, activation='relu'))
modelb.add(layers.Dropout(rate = 0.5))
modelb.add(layers.Dense(1))
modelb.summary()
```

```python

modelc=models.Sequential()
modelc.add(layers.Dense(128, activation='relu', input_shape=(7,)))
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=(7,)))
modelb2.add(layers.Dense(1))
modelb2.summary()

from keras import regularizers
from keras.layers import BatchNormalization
from keras.regularizers import L1L2
reg = [L1L2(l1=0.0, l2=0.01), L1L2(l1=0.01, l2=0.0)]

# LSTM 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()

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))

# Plot validation loss curves
import matplotlib.pyplot as plt

epochs = range(1, len(histLSTM1.history['val_loss']) + 1)
 history_dict_a1_val_loss = histls1['val_loss']
 history_dict_b_val_loss = histls2['val_loss']
 history_dict_c_val_loss = histls3['val_loss']

 plt.plot(epochs, smooth_curve(history_dict_a1_val_loss), 'r-', label
    ='Model A')
```

```
plt.plot(epochs, smooth_curve(history_dict_b_val_loss), 'b-', label=
    'Model B')

plt.xlabel('Epochs')
plt.ylabel('Validation Loss')
plt.legend()
plt.title("Validation Loss")
plt.show()
```

References

Bali, T. G., A. Goyal, D. Huang, F. Jiang, and Q. Wen (2020). Paper the cross-sectional pricing of corporate bonds using big data and machine learning. *The Review of Financial Studies NA*(NA).

Gu, S., B. Kelly, and D. Xiu (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies 33*(5).

Zhang, J. D., X. J. Zhao, J. Zhang, F. Teng, S. Lin, and H. H. Li (2019). Agency mbs prepayment model using neural networks. *International Journal of Theoretical and Applied Finance 24*(4).