Final Report

On May 17, 2019

Can machine learning better forecast Price to Earnings ratios than current methods in small-cap technology sector?

Group 2

Name(UNI):

Ling He (lh2916)
Kitae Kum (kk3281)
Patrick Kwon (yk2805)
Dongoh Shin (ds3709)
Miao Wang (mw3302)
Minsu Yeom (my2582)

Executive Summary

Price-earnings ratio (P/E ratio) is one of the most important tools for equity valuation. Accurately forecasting P/E ratio could help the investors understand which stocks are overvalued or undervalued. For our paper, we focus on forecasting P/E ratios in small-cap technology sector. Our null hypothesis is that machine learning can better forecast quarterly P/E ratios than current methods.

Our data was retrieved from Wharton Research Data Services (WRDS), which provides fundamental values in financial statements such as balance sheets and income statements in a quarterly basis. We collected quarterly data from CRSP (Center for Research in Security Prices)/Compustat Merged Dataset. The data was then further filtered so that it only includes small-cap technology companies (market capitalization within 300 million -2 billion) from January 1970 to the latest recorded date (May 2018). Our original dataset contains 161,852 rows and 34 columns. We later decided to discard data from 1970 to 2005 since we want to avoid over-generalizing our model for periods that do not appear repeated in the future. We also reduced our feature space by handpicking 16 most relevant features.

We used Bloomberg estimated P/E ratios (BEst P/E) and linear regression estimates (LR) as our external and internal baselines, respectively. We chose Recurrence Neural Network (RNN) with Long Short-Term Memory (LSTM) cell as our model because that is one of the state-of-art mechanisms for predicting temporal/sequential data and has shown extraordinary performances in related research projects. One thing worth pointing out is that since our Bloomberg baseline contains about 90% missing values, we used LR estimates to impute them. We consider this combined baseline as our Market baseline. Hyperparameter tuning and early stopping were implemented to ensure that our RNN model would achieve the best generalization while avoid overfitting.

As a result, our RNN model significantly outperformed both baselines according to our chosen metric, Mean Squared Error (MSE). MSE was calculated between the ground truth and the predicted values. Our value-add is a 65% improvement in MSE by RNN model from Market baseline. This provides clear evidence to support our null hypothesis that machine learning can in fact better forecast quarterly P/E ratios than current methods. In addition, our model's robust performance reveals the potential in reducing human labor in decision making process for the investors.

In the future, our work can be extended to forecast more financial variables other than P/E ratios. It is exciting to see how machine learning frameworks such as RNN can effectively improve financial forecasting. We are hopeful that those technologies can lead to advanced process automations, better investment and productivity, as well as enhanced customer experience.

1. BUSINESS UNDERSTANDING

Price-earnings ratio (P/E ratio) is one of the most widely used indicators for investors to make stock selection. By definition, P/E ratio is the ratio for valuing a company that measures its current share price relative to its per-share earnings. It reveals how a stock's valuation compares to its own industry sector. Therefore, accurate P/E ratio forecasts could effectively show whether a company's stock price is overvalued or undervalued. Furthermore, they could help the investors understand which stocks have sufficient growth potential. For our paper, we focus on forecasting P/E ratios in small-cap technology sector.

Our hypothesis is that machine learning can better forecast Price to Earnings ratios (P/E ratios) than current methods in small-cap technology sector for the next three months. We used Bloomberg estimated P/E ratios (BEst P/E) as our external baseline because they are market forecasts. Unfortunately, since there are significant number of missing values in BEst P/E, we decided to use linear regression estimates to impute them. Linear Regression is our internal baseline. Across this report, Linear Regression model Baseline (LR Baseline) refers to a set of forecasted results using linear regression only. We combined BEst P/E with LR Baseline to create Market Baseline, which was first set to be all values of BEst P/Es whenever it is available.

We chose Recurrence Neural Network (RNN) as our model for forecasting. With this setting, we can measure value-add by looking at spreads of Mean Square Error (MSE) between ground truth and each of Model and two Baselines forecasts. We achieved an MSE of 2,409,889.3 for RNN model against ground truth versus 6,850,399.4 for Market Baseline, yielding a spread of 4,440,510.1 provided that we left negative forecasts of P/E ratios out from the final results. The RNN Model's MSE is just 35.3% of that of Market Baseline, leading to an improvement by 64.7%. The Market Baseline includes a total number of 11,639 forecasts, out of which 1,201, or 10.3%, was using BEst P/Es, and we imputed the rest using LR Baseline. In conclusion, our RNN model significantly outperformed Market Baseline.

One thing we want to address is the rationale behind how we dealt with negative P/E ratios. In our models, we discarded negative P/E ratios when we compared our model with the Market baseline. We believe that this decision makes sense because people on the Street take a company with a negative P/E ratio as just one in the red, and no longer consider its P/E ratio as any important. However, we decided to indirectly forecast P/E ratios through predicting the two components, Market capitalization and net income, separately. This allows us to closely inspect the prediction accuracy for either component and understand their impact on P/E ratio.

We took a closer look at negative net income values since they are the ones that result in negative P/E ratios as we know that capitalization is always non-negative. Our results show a MSE of 14,402.27 for all companies including ones in the red. Since LR Baseline has an MSE of 18,079.07, our RNN model achieved a less MSE by approximately 20.3%.

Since our RNN model has shown to reduce MSE for both P/E ratios and net incomes significantly in forecasting them for the next three months, we argue that the null hypothesis holds true. To practically put, our research indicates that we see a possibility that the capital allocated to equity analysts may be more efficiently reduced by replacing some human resources with Model.

Our RNN model performed well in spite of the large skewness of 42.92, which implies its non-normality, of our target, P/E ratio. This result is encourageous since our Model might even generalize better on other sectors than small-cap technology, where distributions are likely to be less skewed. We argue that this advantage is due to our choice of using RNN, which does not require any assumption in a distribution in the dataset used, as opposite to the LR Baseline.

2. DATA UNDERSTANDING

In this Data understanding section, we will explain how we gathered our data from Wharton Research Data Services (WRDS). We then discuss our feature selection and feature engineering process, followed by data pre-processing steps. In the end, we have summarized the changes we made since presentation.

2.1 Data gathering

For our project, we decided to pick stocks from small-cap information technology companies in the USA. For information technology industry, we want to include all companies which their Global Industry Classification Standard (GICS) code that starts with 45, and set market capitalization lower bound (300 million) and upper bound (2 billion) according to small-cap company standards.

Considering those conditions, we have extracted from the CRSP/Compustat Merged Database from WRDS, which provides fundamental values in financial statements such as balance sheets and income statements in a quarterly basis. We also used the US Treasury and Inflation Database, also from Wharton, to observe the CPI index and use it to calculate the inflated value of market cap at that time, to correctly identify the small cap industry of certain years. Starting from January 1970 and until the latest recorded date (May 2018), we subsetted the data set with 16 hand-picked features, in addition to the variables that represent market capitalization and net income. Originally there were 6 other features, but we later deleted them because they were suspected of overfitting the model. We also calculated 3 auxiliary variables, which were calculated by dividing the adjusted price (closing price changed according to the adjustment factor of the company) with the previous value of adjusted price (3 months ago, 6 months ago, 9 months ago). These variables do not directly interfere with the prediction, but generalize the overall prediction and help the model avoid overfitting. The original dataset we gathered has a size 161,852 rows and 34 columns.

We picked small-cap companies to apply to our RNN model, and we researched how to consistently apply a range deciding what is small cap given that we have years of historical data. For example, a $200 million company in 2018 is a small-cap one but it could be a mid-cap back in 1970s. Therefore, we needed a time-variant filtering range to filter small-cap companies to address this time value of money. Our solution was to use Consumer Price Index (CPI). We decided to multiply market capitalization of the companies by (1 + percent change in CPI). In this way, we believe that we appropriately address the time value of money to change nominal values of market caps in the past to real values.

Since company sizes tend to get in and out of what's considered small cap from time to time, there were some missing values in our dataset even though our dataset is known to be well curated. Those missing values were handled mostly through standardization.

2.2 Variable description

(Unit: millions for amount, none for ratio)

Variable Type	**Name**	**Item**
Dependent	Mkvaltq	Market capitalization
	Ibq	Income Before Extraordinary Items
Independent	Saleq	Sales/Turnover
	Coqsq	Cost of Goods Sold
	Xsgaq	Selling, General and Administrative Expenses
	Oiadpq	Operating Income after depreciation
	Cheq	Cash and Short-Term Investment
	Rectq	Receivables
	Invtq	Inventories
	Acoq	Other Current Assets
	Ppentq	Property Plant and Equipment

	Aoq	Other Assets
	Dlcq	Debt in Current Liabilities
	Apq	Account Payable/Creditors
	Txpq	Income Tax Payable
	Lcoq	Other Current Liabilities
	Ltq	Long term Liabilities
Auxiliary	Mom3m	Ratio between adjusted price (closing price / adjustment factor) for 3 month difference
	Mom6m	Ratio between adjusted price (closing price / adjustment factor) for 6 month difference
	Mom9m	Ratio between adjusted price (closing price / adjustment factor) for 9 month difference

Table 1 : A variable list

For each stock and at each quarter, we have 16 features for inputs from items in financial statements, in addition to 3 auxiliary features. Also, we used Market capitalization and Income Before Extraordinary Items (net income) to calculate P/E ratios.

Income statement items are quarterly results, denoted q as suffix in variables names, and balance sheet items are results as of the end of each quarter. The income statement items include sales, cost of goods sold (COGS), selling, general and administrative expenses (SG&A expenses), and operating income after depreciation. The balance sheet items are cash and short-term investment, receivables, inventories, other current assets, property plant and equipment, other assets, debt in current liabilities, account payable, income tax payable, other current liabilities, and long-term liabilities. They are all quant-type variables. A full list of variables is in Table 1 grouped by independent and dependent ones. Since we use historical data, all data is fixed. Most of data are time-series. Our RNN model are inherently designed to take such a sequence of data, whereas the baseline model, linear regression, should assume independence.

We conducted research to select each feature by hand first among 674 variables available in WRDS. As a principle, we tried to comprehensively cover essential flows from the top line to the bottom in a income statement, while we didn't want to begin with too large set of redundant variables. The four income statement items are directly connected to earnings, one of our dependent variables. Sales is the initial item for forecasting earnings. It serves as the top line for measuring earnings so it provides us with a means to guess the amount of earnings although it alone does not give a clue of the exact profitability. COGS and SG&A expenses determine a firm's profitability given its sales. We believe that COGS is more likely to hold more information about profitability than SG&A expenses. The latter usually includes one-off items more often than the former does. COGS include the cost of products or raw materials and direct labor for workers who participate in producing the products. SG&A expenses indicate all the costs not directly tied to making a product or performing a service such as R&D expenses, rent, and advertising expenses. Operating income after depreciation is the more direct item to forecast earnings than the other three. If we add non-operating income, such as gains and losses from investment and currency exchange, to the item, we can obtain earnings directly. We believe that it is a key for forecasting earnings.

We applied the same principle to handpick items in a balance sheet as well: comprehensive, but not very redundant. We decided to allow some degrees of overlap because we didn't want to end up with too small set of data. For example, both of debt in current liabilities and account payables are under the current liabilities item, but both are picked as features. It is because we want our RNN model, not handpicking, to decide which one is more influential during a training process. There was no clear cut, but we wanted to be careful about this. This would impact not only on the training time, which was a unit of hours for our RNN model, but also on a final performance of the model.

Balance sheet items plays a rather supplementary role for forecasting earnings. We use cash and short-term investment to find a relationship between earnings and cash flows. Short-term investment indicates an investment of five years or shorter, for example, savings account, CDs, money market accounts, and treasury bills. Earnings are divided into two parts, which are non-cash earnings and cash earnings. Since cash earnings are converted to the item, change in cash help predict earnings before extraordinary items. Receivables and inventories are factors of working capital. They are related to earnings which are not yet converted to cash. With change in cash and short-term investment, they have an explanatory power about earnings. Other current assets are a category of current assets that does not include cash, securities, receivables, inventories, and prepaid assets. We believe that they are not as associated with earnings as the three balance sheet items mentioned. But they are convertible into cash within a year so it is still important to have them as our features.

Companies allocate their capital in Property plant and equipment (PPE) in order to generate future economic benefits. As PPE is one of the largest and expensive tangible assets to acquire, they make their decisions carefully and comprehensively. Therefore, a significant change apart from depreciation/amortization would be a good signal for a subsequent change in future earnings. Other assets are non-current assets that do not belong to any category of investments, PPE, or intangible assets. We believe that they have a less predictive power, so we do not classify them further into finer items. Still, we intend to include them as a feature in order for our model to decide how influential they would be, leaving a possibility of an unexpectable contribution to forecasting. We expect that debt in current liabilities would help forecasting in a opposite way. Specifically, higher debt in current liabilities, alone or probably combined with others such as cash and short-term investment, and/or inventory, indicates a unhealthy financial status. We believe such a risk factor functions as a regulator in forecasting so that it is not too optimistic.

While account payable is also one type of short-term debt, we include it because we think that a sudden big change in value of this item may imply a huge increase in demand so that companies need more supplies. Income tax payable is a tax that has not yet to be paid. We hope that our model classifies an earnings surprise accompanying with an increase by exactly that amount as one-time event. The last two items in balance sheet we used are current liabilities and long-term liabilities. Excessive size of liabilities relative to companies' equity would be problematic in generating future earnings due to the costs. Therefore, we believe that taking them as features would have our model against companies with large increases in liabilities, or vice versa.

2.3 Feature selection & Feature Engineering

2.3.1 Feature selection

Features we selected were discussed in 2.2 Variable description. More specifically, we used cumulative values for features from income statements. We believe that past 12 months is the standard period of time used in the financial statement, so that we adapt it in our feature selection. As for features in balance sheets, we used the most recent values available at a point of time they are used. We selected stock prices as one of features as well in a hope that this would address company-specific events and market regime changes. Obviously, the frequency of all other features is quarterly so that we believe the price features would compliment this lower frequency.

It is not necessary to perform dimensionality reduction since we used RNN as our model. One advantage of any neural network is that they inherently learns feature representation.

2.3.2 Data standardization

We standardized our features for both LR Baseline and RNN model. While we only used Robust Scaler for preprocessing in LR Baseline, we implemented a more complex standardization process for RNN model.

Each feature has a very different scale. Therefore, it is necessary to normalize all features so that we have zero mean for the following two reasons: 1) efficient convergence speed, and 2) fair comparison. Any neural network expects the dataset to be normalized so it would lead to a reasonable convergence speed. As for the fair comparison, if a company size is drastically different from the others, then most of the fundamental variables we used would also be proportionally different in size. For example, the amount of sales of a $2,000 million company in market capitalization would have much larger net income than that of a $300 million company. In such case, we can't do an apple-to-apple comparison. Thus, normalization is much needed.

The next step to consider is to pick an appropriate normalizer. We chose market capitalization while we could have picked another such as book value. The reason was that sometimes we found companies with negative book values, whereas market capitalization can't be negative. Specifically, for this normalization, we divided numerical features such as Sales, COGS and Other Assets by the lastest market capitalization at each point of time. To calculate the market capitalization of each company, we used Closing Price * Common Share Outstanding. It was suggested at first to impute Mkvaltq_ttm, or market capitalization, but chose to apply to all companies that already have market capitalization filled for consistency.

Also, we used the natural log to counter large values. Lastly, we applied Robust Scaler on all the preprocessed values.

2.3.3 Feature engineering

While most of our features are not engineered from the original values to another apart from standardization, stock prices are used as percent changes. We name them momentum variables. They were acquired by dividing stock prices by the adjusted price variables according to different period of times such as three, six and nine months.

2.4 Changes

As we develop this project, we noticed that we need to make some changes in our original plan. We decided to train our model using more recent data, and discarded data from 1970 to 2005. The rationale behind this decision that we don't want to over-generalize our model for periods that do not appear repeated in the future. Thereby, we are able to deliver a smaller prediction error measured by MSE. This is particularly plausible as we are dealing with small-cap technology sector, where disruptive changes have happened drastically in the past several decades. Specifically, continuous training indicated that the data trends from 1970 to 2006 would harm the MSE between the prediction and the ground truth. Thus, the dataset was modified to only contain the values starting from 2005/10, which were done by filtering on the "active" variable, an indicator to living companies.

In addition, the following variables were no longer used as features because they barely contributed to the model in terms of feature importances: Total Parent Stockholders' Equity (seqq_mrq), Total Assets (atq_mrq), Number of Common Shares Outstanding (csho_1yr_avg), Adjusted Price (adjusted_price), Close Price (prccm), and Adjustment Factor (ajexm).

3. MODEL UNDERSTANDING

In this Model understanding section, we discuss the rationale behind our model selection. In addition, we observe the statistics of our dependent variables and carefully examine the variable relationships in our LR Baseline and RNN.

3.1 Model Goal, Model Selection

The goal of our model is to derive better forecasts than the two baselines. To achieve this goal, we chose RNN as our model to generate the largest value-add. The reasons are twofold: 1). both dependent and independent variables have typical characteristics of sequential and repetitive data, and 2) RNN automatically learns impactful feature representations during training as any deep learning model can approximate universal functions.

The choice of cell type and optimizer are crucial to any RNN models. After some experimentation and literature review, we decided to use the Long Short-Term Memory (LSTM) cell rather than the Gate Recurrent Unit (GRU). In theory, LSTM controls the exposure of memory content while GRU exposes the entire cell states to other units in the network. In the case of our data, which has long-term dependence, we set up a hypothesis that LSTM has higher accuracy. Furthermore, during hyperparameter tuning, it is proven from the comparison of the predicted values of GRU and those of LSTM that LSTM provides higher accuracy from 3rd decimal points with our target data than the GRU.

As for the optimizer, we chose Adadelta as our optimizer from the collection of optimizers in Tensorflow. It does not only have the fastest recurrent networks but also incorporate adaptive learning rate, which decays in a linear fashion and automatically updates a fixed time period of data. Other important hyperparameters such as batch size and learning rate are also tuned during our training phase.

Regarding implementation details for the RNN model, we coded our main program using Tensorflow because of its wide usage and adaptability. During the process, the dataset was divided into many ‘batches’ depending on company ID and dates. Each batch contains approximately similar number of rows so that they will generally have the same dimension.

The batches were scaled using the Robust Scaling system, provided by the python package sklearn. Each scaling parameter was stored in memory in order to be used for converting the predictions back to original scale. In addition, we splitted the batches into training and validation sets, where the training set contains 90% of the batches and the rest 10% are in the validation set. For each training epoch, the model will collect the training and validation MSE generated from the epoch. If the model starts to have very little marginal improvement on the validation MSE after certain number of epochs, the model stops and generates the model based on the minimum validation MSE. In fact, this mechanism is called early stopping in order to avoid overfitting.

After we trained our RNN model on each of the two components of P/E ratio, market capitalization and income before extraordinary items, we generated quarterly P/E predictions for companies included in our data. Unlike traditional machine learning methodology where we split our data into training, validation, and test sets, we actually use the same dataset for training and test. Due to the sequential nature of our data, the past quarter's feature values can be used to predict the present/current quarter's P/E ratio, and the present/current quarter's features values can be used to predict next quarter's P/E ratio.

We encountered several obstacles in the process of model development. One of them is that some companies may become inactive on the market due to bankrupt or delisting over some period. Therefore, as mentioned in previous section, we handled those discontinuous data by filtering out the rows where the companies are inactive indicated by the variable "active".

Another problem we tackled is the negative P/E ratio problem. Originally, P/E ratio is calculated by dividing Market capitalization by Income before extraordinary Items (net income). Therefore, it is natural that the value of P/E becomes negative or infinity when net incomes are negative or zero respectively. In this case, P/E ratio becomes meaningless. Thus, we decided to predict the two components of P/E ratio, market capitalization and the net income, separately using RNN. This allows us to prevent encountering those invalid P/E values that may lead to a distorted result. In the end, we calculate predicted P/E ratio from those the predictions of the two components.

3.2 Descriptive Statistics

3.2.1 Statistics on P/E

We summarize the primary statistics of our target (ground truth), our model, and the baselines on the table as below. According to the table, we can argue that our RNN model suggests good forecasts now that the extent of dispersion from the mean is similar to that of our target. The target's standard deviation is 781.745, which is the lowest and followed by RNN estimates, 1,453,913. Besides, our model and target are also similar in terms of the shape of the distribution. Our target is skewed to the right since we get rid of all the rows with negative P/E of the target. The only distribution which is positively skewed is our model's, whereas those of the other two are heavily skewed to the left. This is summarized in Table 2.

	25%	**Median**	**Mean**	**75%**	**Std dev**	**Mode**	**Skewness**

Target	15.81	24.09	81.22	41.97	781.75	91.24	42.92
RNN	-5.85	14.94	11.63	26.90	1,453.91	19.94	11.60
LR Baseline	17.69	26.84	22.41	46.71	2,421.08	90.27	-93.64
Market Baseline	16.52	25.08	20.74	42.12	2,412.45	10.10	-94.61

Table 2 : Descriptive statistics on P/E ratios

Before examining mean and quartiles, we need to clarify how we preprocessed one of our dependent variables, income before extraordinary items. We basically multiplied each of them by four to annualize since we have only quarterly earnings for our original data. Bloomberg Estimated P/E is an annualized consensus so we leave the original as we download it from the source.

Our target has its median of 24.09 and mean of 81.22. We believe that the reason of the notable difference between the mean and median is because we truncate negative P/Es of the target. The difference of mean between the target and the other three is too big to judge which model provides similar estimates to ground truth. On the other hand, linear model and the integrated baseline provide similar medians to that of the target, which are 26.84 and 25.08, respectively.

3.2.2 Statistics on Net Income

The portion of companies that have either negative or zero net income is about 35.7%; the total number of the companies is 19,407 and 6,944 net incomes were negative. For those companies, calculating a valid P/E ratio is not available. Therefore, our forecasting of P/E ratio is processed within the constraint of its positive values. In fact, in the case of earning negative values on P/E ratio, analysts in real fields also skip analysis on those values.

However, even if net income is negative, they are still meaningful. Since net Income is negative, the company is in a deficit, so we can provide analysts with an indicator of the attractiveness of the company by forecasting at the point at which the net income changes from a positive number to a negative number.

From this perspective, by separately analyzing the negative net income, which is not reflected in the P/E ratio, we help Analysts to reduce resources for verifying their investments under deep-in-the-money or deep-out-of-the-money.

3.3 Functions types, variable relationships

3.3.1 LR Baseline

In LR Baseline, our function type is Y=f(X,B) as we assume a linear relationship between our independent and dependent variables. We examined the coefficients of our covariates to gain insights on feature importance. Since we used robust scaler in data pre-processing, the values of the variables are on similar scales. Therefore, it is reasonable to believe that the magnitude of coefficients reflect covariates' impact on the dependent variables.

The top three most influential features on market capitalization prediction are: Long term Liabilities, Other Assets, and Sales/Turnover. While Long term Liabilities can negatively be correlated with market capitalization, the other two would have a positive impact. It is not surprising to see that more Sales would lead to an increase in market capitalization, but Long term Liabilities has the opposite effect. In addition, it is quite interesting that Other assets has more predictive power on the market capitalization than expected.

Similarly, for the prediction of Income Before Extraordinary Items, the top three most influential features are: Operating Income after depreciation, Long term Liabilities, and Other Assets. Only Long term Liabilities has a negative impact on the response while the other two are positively related with the response. As expected, Operating income after depreciation has the most influence on our dependent variable since it is an obvious indicator of Income Before Extraordinary Items. The other two important independent variables have similar effect on Income Before Extraordinary as to market capitalization.

Two out of the top three most significant features are the same for our two dependent variables, market capitalization, and Income Before Extraordinary Items. This indicates that the two parts that constitutes P/E ratio are inherently homogeneous. Furthermore, inspection of feature importance also helps us to have a more thorough understanding of what factors contribute to which specific part of P/E ratio.

3.3.2 RNN

In RNN, the output Y is a nonlinear function of the weighted sum of the activations of all units that connect to it through all the layers. It is known that the interpretability of deep learning models is relatively weak compared to other machine learning frameworks. The nonlinear nature of neural networks makes them hard to understand. This situation is even more challenging for RNN since it incorporates recurrent internal state updates and temporal sequence.

Therefore, it is difficult to examine feature importance in our RNN model, and we have decided to include it in our future work. In fact, there is an active research community within this field that works on developing RNN models that focus on both temporal aspect of data and characterizing feature importance.

4. MODEL PERFORMANCE

In this model performance section, we discuss the hyperparameter tuning process for RNN in order to achieve the best possible model. We then evaluate our model performance in three perspectives: 1) Overall intra-model performance comparison of Market baseline, LR baseline and RNN model using MSE, 2) Inter-model comparison of LR baseline and RNN model, and 3) Business case study of a sample small-cap technology company. Lastly, we address the various issues that we have overlooked.

4.1 RNN tuning

In order to achieve the best performance, we performed hyperparameter tuning on our RNN model. As shown in Table 3, our tunable hyperparameters include cell type, optimizer, number of layers, number of units in each hidden layer, drop out rate, and batch size. We experimented architectures with various values of the hyperparameters, and evaluated the results by Mean Squared Errors (MSE). The MSE is calculated between the ground truth and the predicted values generated by different hyperparameter values.

Hyperparameter	**Optimal parameter value**
Cell type	LSTM
Optimizer	AdaDeltaOptimizer
Number of layers	3
Number of units in hidden layer	128
Learning rate	0.6
Probability of keeping unit during dropout	0.75
Batch size	128

Table 3 : Hyperparameters for Model

We first kept the other parameter values constant and experimented various cell types and optimizers. Specifically, we tried Gate Recurrent Unit (GRU) and Long Short-Term Memory

(LSTM) since they are the mostly commonly used cell types in RNN. The MSE of LSTM is slightly better that of GRU. Also, the LSTM's property of having 3 gates, which is one more than GRU's, enables us to have more control over the data to be trained.

Similarly, we tested AdaDeltaOptimizer and RMSPropOptimizer, which are two popular optimizers provided in Tensorflow. Our results show that AdaDeltaOptimizer significantly outperformed RMSPropOptimizer.

For the other hyperparameters, we evaluated 1, 2, and 3 hidden layers, 64, 128, and 256 units for each hidden layer, as well as various learning rate, drop out rate, and batch sizes.

We also implemented early stopping in our RNN model in order to prevent overfitting, which is a common problem in many deep learning frameworks. We devised our program so that it stops training when there is no significant improvement on the validation MSE after 10 epochs. Our validation dataset includes 10% of the data. For each epoch, we recorded the training and valid MSE's in order to understand model performance and evaluate our model more thoroughly, and created graphs to observe the decreasing trend.

Our RNN model was trained twice, once for each of the two components of P/E ratio, market capitalization and income before extraordinary items. After we obtained the predicted values for each component, we then calculated our predicted P/E ratios from those them. For the calculation for market capitalization, the model stopped early after 50 epochs. For the calculation for income before extraordinary items, the model stopped at 80.

4.2. Performance comparison

4.2.1 Overall comparison

Metrics	**Market Baseline**	**LR Baseline**	**RNN**
MSE (P/E)	6,850,399.4	6,861,625.7	2,409,889.3
MSE (negative net income)	N/A	18,079.07	14,402.27

Table 4: Overall MSE Comparison and R-square results for LR

In order to evaluate how good our models are, we want to determine how accurate our predictions are. Thus, we want to analyze the difference between the predicted and the actual value from historical data, i.e. ground truth. A smaller difference implies a better accuracy. Because our results are numerical, we chose to use Mean Squared Error (MSE) as the metric to calculate the accuracy of the prediction. It is a commonly used general purpose quality estimator. MSE measures the average magnitude of the error and ranges from 0 to infinity. The errors are squared and then they are averaged. MSE gives a relatively high weight to large errors, and the errors in predictions can be critical, so it is an appropriate metric to penalize the large errors.

In Table 4, we calculated MSE of P/E ratio predictions for all three models which are Linear Regression baseline (LR), Market base line (BEst + LR for imputed values) and RNN model. From our results, we can see that MSE(RNN) < MSE(Market) < MSE(LR). Therefore, from the MSE perspective, our RNN model significantly outperformed both internal and external baselines. Our results displayed a statistically significant advantage over the baseline, since our value-add is a 65% improvement in MSE by RNN model from Market baseline.

In addition, we discussed that it is also worth looking into negative net income because they still contain certain information while negative values in the context P/E ratios are meaningless. For MSE of negative net income, we also got MSE(RNN)< MSE(LR). Our RNN model decreased MSE by approximately 20.3% compare to LR Baseline.

These arguments provides strong evidence to support our null hypothesis that machine learning can in fact better forecast quarterly P/E ratios for the next 3 months than current methods.

Another thing we want to address is that originally, we also planned to compare our model with the baselines using an additional metric, R-squared. However, we decided later that R-squared would no longer be used. One reason why we decided so was, to directly express how much value-add our models provide, that we should have a comparable metric between our models, i.e., a recurrent neural networks, and baselines, which is a linear regression. R-squared did not help us in this front. Thus, we only include R-squared for LR Baseline.

4.2.2 Inter-model comparison of LR Baseline and RNN model

Metrics	**LR Baseline** (Market Capitalization)	**LR Baseline** (Income Before Extraordinary Items)
R squared	0.65	0.35

Table 5: R-squared for Linear Regression baseline

We first examined R-squared for LR Baseline. R-squared explains how close the data are to our fitted regression line. From table 5, we can see that for market capitalization, our R-squared is 65%. And for net income before extraordinary items, R-squared explains 35% of the total variation. Generally, 65% is not high enough to conclude that it fits well our data. LR Baseline is, therefore, not a desirable model for forecasting market capitalization, net income, and thus P/E ratio.

Next, let’s move to our value-add model RNN. We plotted Train MSE versus Epoch and Valid MSE versus Epoch for net income and market capitalization. The values on the y-axes are different from the original scale due to the scaling procedure implemented in pre-processing for training. However, we later converted the predicted values back to their original scale during testing.

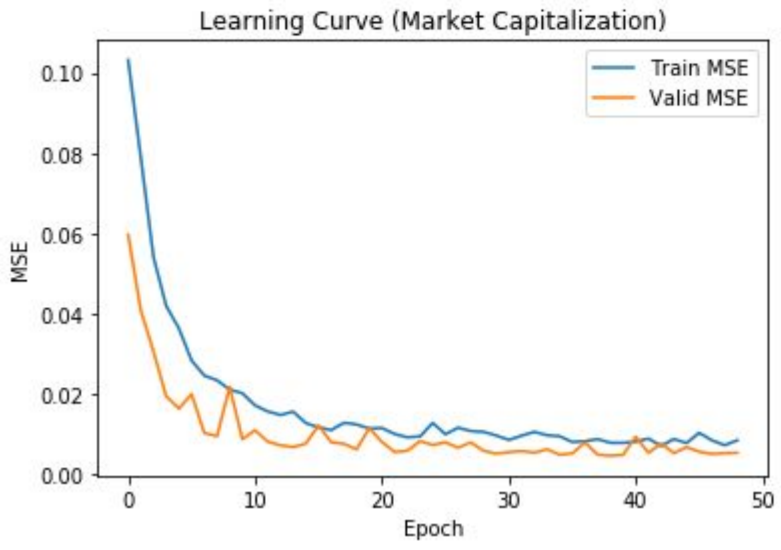

Figure 1: Learning Curve (Market Capitalization)

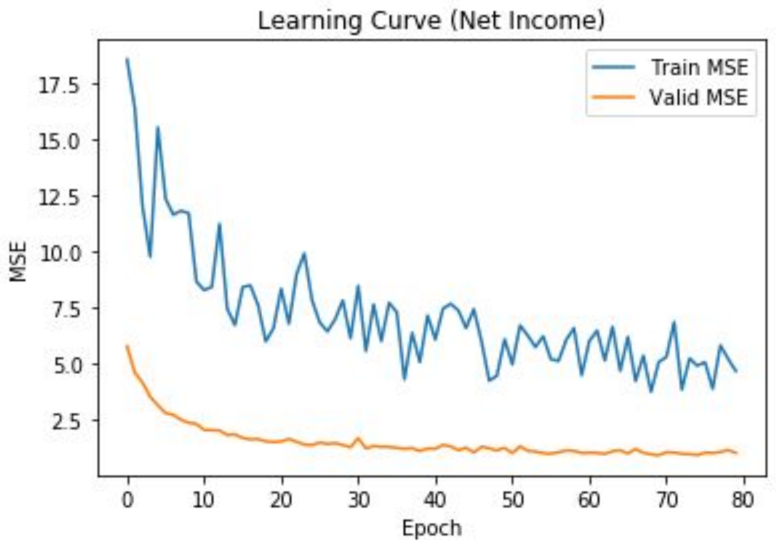

Figure 1 shows the market capitalization MSE plot. The final valid MSE for market capitalization was 0.0048, which is only 8% of the MSE at the first step, and it was reached at the 40th epoch.

Figure 2: Learning Curve(Net Income)

Similarly, the final MSE at predicting the income was 0.9388, which is only 16% of the MSE at the first step. Also, the final MSE was reached at the 75th epoch. Both the train MSE and the valid MSE displayed a relatively smooth decreasing pattern, with no abrupt spikes. Furthermore, for both plots, Train MSE fluctuates more than valid MSE, which is a good sign. This shows that our model is stable. Our results demonstrat a statistically significant advantage over both of the baselines

4.2.3 Business case study

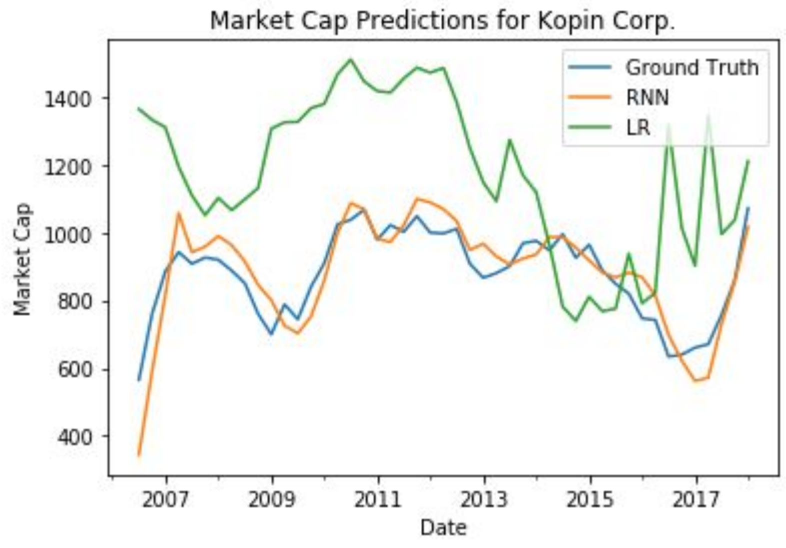

Figure 3: Market Capitalization Prediction for Kopin Corporation

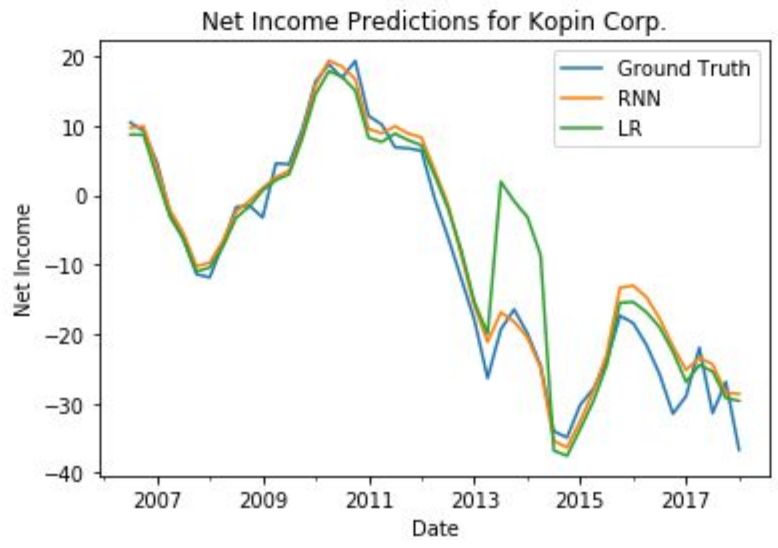

Figure 4: Net Income Prediction for Kopin Corporation

In order to explicitly demonstrate the business value-add of our RNN model, we selected a sample company, Kopin Corporation (KOPN), and plotted its forecasts. Kopin Corporation (KOPN) is a Massachusetts based electronic manufacturer.

In either Figure 3 or Figure 4, Our RNN model shows a much more similar trend to the ground truth compared to LR Baseline. This allows us to explicitly observe that the MSE of RNN is much smaller than that of LR Baseline for any of the dependent variables. In particular, RNN significantly outperformed LR baseline in market capitalization for this specific company.

(Unit : $1,000)

Year	2016	2017	2018
Revenue	4,041.43	6,358.41	3,741.50
Expenses	-27,475.54	-31,598.90	-39,611.12
Net Income	-23,434.12	-25,240.48	-35,869.63

Table 6: Income Statement of Kopin Corporation from Yahoo finance

We also retrieved this company's data from Yahoo finance. There are several interesting observations. KOPIN is one of the companies that have negative net income values, which we have discussed previously because they could lead to negative P/E ratios. Also, the numbers from Yahoo finance seem to agree with the trends in our graphs.

From this promising example, in the future, we can expand our model to predict other financial variables of Income Statement, not only P/E ratio. This real-world business case above gives a glimpse of how machine learning frameworks such as RNN can effectively forecast financial variables. Therefore, through these technologies, we can expect automation of analysis of complex financial indicators. In addition, analysts can expect to improve their investment mechanism and productivity thereby providing more accurate consulting to their clients.

4.3 Issues overlooked

4.3.1 More critical model selection

We acknowledge that we were overconfident with RNN although we thought we had a good reason to draw this conclusion. As we work with time-series data, there seemed not to be a better choice than RNN. However, looking back, we could have done more comprehensive comparisons with other models including neural networks such as Multilayer Perceptrons (MLPs). For example, we could have very simplified this problem and try other models with a very simple data set. By that way, without spending too much time and efforts, we would insist that we went through a more entrusted process of model selection. After we started to work on the real data set, which takes 2-3 hours to train, it was very difficult to even discuss going back from the model comparisons.

4.3.2 Missing values in BEst P/Es

We chose a small-cap sector for this project because we thought the small-cap market is less efficient than the large-cap market, giving us a better chance of achieving our goal, to forecast better than Market Baseline. However, we did not realize that Bloomberg BEst P/Es have much more missing values in the small-cap markets than in the large-cap market. In this specific market and sector, we have 90% missing values in the market forecasts. This could, therefore, give us a biased comparison, and possibly affect on the model performance. As we had announced this topic from the very early in this semester, it seemed quite late to change it during the semester. We want to admit that we should have recognized and raised this issue earlier. Then, we would have a more unbiased and more confident result.

4.3.3 Higher frequency of data

As most of our data come from financial statements, we have to have quarterly frequency of the data. This implies less data points, and possibly a larger bias in our data set. We were concerned about this, and tried to find a way of improving it. One suggestion was just to do a linear interpolation between quarters to change quarterly data into monthly data. However, it just infuse artificial, linear data into the genuine one, so we did not choose this idea. This lower frequency still seems to be an inevitable choice, but we could have tried to build a very short-term model to get a higher frequency dataset.