Math642 Introduction to Machine Learning  
Spring 2020

Final Project

Fyona Sun

Introduction

Online auction has been one of the most popular methods to buy and sell items online. It is part of the growing body of eCommerce system and can be further apply to other competitive markets. EBay, Yahoo! Auctions, Amazon Marketplace have started this auction methods a long time ago, recent year there are new sites who focus on creating a second-hand and sustainable environment also adopted this bidding system. The activities of online auction produce a large number of bidding histories and transaction data that can be utilized to provide insights into e-commerce system.

Forecasting the auction outcome is beneficial to all auction parties. Bidders can use price forecasts to make more informed bidding decisions based on their budgets. Since it’s difficult for a bidder to monitor all the active auctions, it might be difficult for a bidder to decide which of these numerous auctions to participate in and to place a bid that does not end up overpaid or failed to win the auction.

Sellers can use price forecasts to determine when to post their listing. There are companies that re-sell materials on eBay could also make use of price forecasting. These stores sell materials on behalf of individuals who do not want to use eBay directly. Ongoing price forecasting would allow these stores to pay their customers for their merchandise before the close of the auction, in effect offering faster liquidation for their customers.

Online auction platform providers and auction houses can use forecasts for long-term budgeting and planning purposes or even real-time adjustments during the bidding process. Another way to think in the online setting is that a dynamic price scoring would allow auctions to be ranked by lowest expected price, which would in turn allow purchasers to focus their time and energy on just those few auctions that promise the lowest price. Thus, a well-functioning forecasting system could be adaptive to accommodate and incorporate change in the dynamic setting.

The study of the mechanics of online auction is not only useful in respect to the existing online auction platforms, but also can be applicable to other emerging markets such as search driven online advertising business. By 2018, more than 80% of digital display ads are bought in the form of real-time bidding. Thus, understanding the challenge and of online auction data provide practical solutions for these problems may help us design a better and more efficient programmatic bidding system.

Problem Description

Modelling and predicting the bidding price of online auction in real-time can be a challenging task. The price dynamics change throughout the auction and directly reflect the bidding behavior of all parties participated.

First of all, the online auction data has a relatively complicated structure. The dataset contains two different types of data, the bid history and the auction related information. The bid history is a sequence of bids placed over time, which can be considered as a time series type of data. One the other hand, the auction related information is a cross-sectional type of data. It contains product information, information about the bidders and the sellers and auction type information that does not change during the auction. Thus, creating a model that can capture both temporal and static information is challenging.

Moreover, most of the time series methods typically assume that events arrive at equally spaced time intervals, for example in stock market, the timestamp of price is placed equal length. However, in the case for online auction data, the timestamp is generated by user activities, where events are not equally spaced. Therefore, each auction has different staring times, different ending times, different durations and different bidding decisions are recorded. This is the first time I tried to handle a dataset like this.

The main challenge I encounter in this project is defining the machine learning object of my model. Although it’s our initiation to predict the end-price of an auction, which can be done easily using given information on its category, starting bid value and so on, the real challenge here is how can we predict the future price with ongoing information. In this project I am trying to build a model is capable for real-time predicting along with the auction. I will address more on this issue in the next section.

Objective and Methods

The main goal is to develop a model that can predict the auction outcomes, that is the auction price at a future time (T+1) using information from the present (T) and the past (T-1, T-2,…,1). Assuming the estimated model parameters did not change from time T-1 to time T, at time T (present), the model can make a prediction about the future price at time T + 1.

First I consider static models (information given before the start of the auction) that predicting the end-price of an auction using multiple linear regression, KNN and XGboost. Then I have generated some new variables to capture the price dynamics, using the information at present time T to predict the future time T+1. The final approach involves information from a period of time (T, T-1, T-2 etc.) by implementing LSTM neural network to predict the future price(T+1).

Since predicting future price is a regression problem instead of a classification problem, all model performances are evaluated by MAE (Mean Absolute Error).

The smaller the mean absolute error, the better the model predicts.

Data Description

The original data set is from Kaggle (https://www.kaggle.com/onlineauctions/online-auctions-dataset), which was collected from Ebay for the book Modeling Online Auctions, by Wolfgang Jank and Galit Shmueli (Wiley and Sons, ISBN: 978-0-470-47565-2, July 2010). The dataset includes 10681 observations and 9 variables.

|  |  |
| --- | --- |
| auctionid | the unique identifier of an auction |
| bid | the proxy bid placed by a bidder |
| bidtime | the time in days at which the bid was placed (from the start of the auction) |
| bidder | username of the bidder |
| bidderrate | bidder's rating |
| openbid | the opening bid (set by the seller) |
| price | the price that the item was sold for (equivalent to the second highest bid + an increment) |

It includes the complete bidding records for 627 auctions on Cartier wristwatches, Palm Pilot M515 PDAs, Xbox game consoles. The auction durations vary from 3 days to 7 days. While analyzing the original dataset, I found that the last bid is not necessarily the highest bid, nor the last bidder is necessarily the one who win the auction. It's important to understand how eBay auction policy works before conducting further analysis.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA picture containing screenshot

Description automatically generatedA picture containing screenshot

Description automatically generated

People have to bid higher than current price, which is the second highest bid increment. If someone places a bid much higher than current price, eBay will only use part of his/her bid (second highest bid increment) as the new current price. People can bid as long as their bid is higher than current price. Therefore, as bid time increases, we sometimes see bids that are lower than previous bids. People will be notified when they are outbid, so they may increase their max bids to remain the higher bidder. So the real-time price one sees on the website do not directly reflect the latest bid, and it could be misleading to using the bidding value as the real-time auction price.

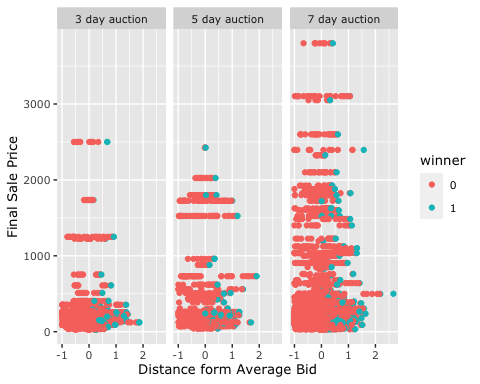
To simulate the real-time auction price, I create a function following the logic of EBay auction displaying policy and transformed a sequence of bidding values to the sequence of ‘current price’ that could have been displayed on the website. The method is described in the book *Modeling Online Auctions*.

Since the auctions has different auction durations, I normalized the bid time by dividing the bid time over the auction length. So that the sample auctions are on the same time scale from 0 to 1. This also makes it easier to add features such as the bid distance from the open bid and the bid distance from the average bid.

For each auction, I add the number of bidders, average bidding value, the standard deviation of the bidding value, the bid time difference and the bid value difference between each bid and the price velocity to capture the on-going features of an auction procedure.

Finally, I transformed the characteristic variables to dummy variables to keep the text information. The dataset after the data manipulation and cleaning the outliers, turns out to have 10677 observations and 30 variables. The distribution of the end-price is affected a lot by the product type, the distribution of the frequency of the bid time increases drastically for all auctions despite the different auction lengths. As I have explained before, the last few bids are not necessarily the winning bid. However, looking at the distribution of the winning bids, we can find that the winning bids are most likely to appear at the last 75% of the auction duration.

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Unsupervised Modeling of Online Auction

1. Modeling Individual Auctions

The plot shows an example of the raw bidding history of a sample auction and the recreated livebid of the same auction. We can see that the livebid is a non-decreasing curve that represents the auction price that was displaying on the auction website during the auction. Since the for the auction data set the timestamp is generated by user activities, where events are not equally spaced, my intension here is to use the unsupervised modeling methods to turn a discrete time series to a continuous function so that I can recreate equal length time series for each auction. In this section I applied polynomial function, regression spline, smoothing spline and GAM to model an individual auction.

A picture containing photo, game, man, table

Description automatically generated A close up of a person

Description automatically generated

1. Polynomial Fits

I applied the polynomial fit for the current price using the bid time. The RSS decreases as the degree of the polynomial decreases. It achieves its minimum at degree 7. However, it diverges as the degree of polynomial increases. Thus, from the plot we choose the degree of 3. All three degrees are significant with a p-value of 0.0001607.

A close up of a mans face

Description automatically generatedA close up of a map

Description automatically generated

1. Smoothing splines

We want to find some function that fit a smooth curve to the set of data by automatically choosing the smoothing parameter λ. Here I applied smoothing spline to this sample data for a range of degrees of freedom and plot the resulting fits. RSS decreases as the degrees of freedom increases at 6. Then I select the smoothness level by cross-validation; this results in a value of λ that yields 7 degrees of freedom. The regression fits the data well with a p-value of 0.0005362.

A close up of a map

Description automatically generatedA close up of a map

Description automatically generated

1. GAM

The method of GAM allows us to fit a non-linear model to the set of data, so that we can automatically model non-linear relationships that standard linear regression will miss without manually try out many different transformations on each variable individually. The GAM fit is said to be more flexible than the Splines. Here for this sample set it returns a flexible curve with 11 degrees of freedom, which might results in overfitting.

A close up of a map

Description automatically generated

1. Modeling Multiple Auctions

Since the auction dataset does not provide me with an actual time, for my study I assumed that these auctions happened simultaneously so that I can use all of the auctions as my sample data. I normalized the bidding time and the auction length so that all of the 628 auctions can be displayed on the same time scale. I fit the live bidding history for each auction with the unsurprised models I described above and classified the 628 models into 4 groups based on the smallest SSE. The method has been discussed in detail in the Chapter 4 of the book *Modeling Online Auctions*. I could have applied k-nearest neighbors but since creating a distance matrix for a time series is relatively challenging, I think I could complete this idea in the future work.

A screenshot of a cell phone

Description automatically generated

table(models)

## models  
## 1 2 3 4   
## 301 5 86 212

Supervised Modeling of Online Auction

1. Predicting the end price with Static Model

In this section I could discuss how I build models to predict the end price of the item would be auctioned for. I implement machine learning methods such as Multiple Linear Regression and KNN.

Since the auction dataset is sequenced by individual auctions and bidding history for each auction, random sampling for the entire observations does not make sense. Thus, I grouped bidding history information by auction id and split the 627 unique auctions into training and testing set. We have 369 auctions with 7557 observations in the training dataset and 258 auctions with 3120 observations in the testing dataset.

For the baseline model I use the static features, that is the information are given before a certain auction starts and do not change during the auction. The static information includes:

- opening price of the current auctions

- duration of the current auctions

- sellers’ ratting of the current auctions

I trained the multiple linear model with the training data and 10-folds validation method.

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | Rsquared | MAE |
| Training | 268.2168 | 0.5408452 | 132.4348 |
| Testing | 303.4196504 | 0.3509475 | 184.0715687 |

For the KNN model, the k was chosen automatically by the cross-validation. RMSE was used to select the optimal model using the smallest value. The final value used for the model was k = 5.

|  |  |  |  |
| --- | --- | --- | --- |
| k | RMSE | RSquared | MAE |
| 5 | 203.5048 | 0.7333769 | 76.95662 |
| 7 | 208.5114 | 0.7198521 | 82.10603 |
| 9 | 217.1711 | 0.6960207 | 87.56857 |

A close up of a map

Description automatically generated

With k=5 the KNN model prediction on the testing set gives RMSE of 95.92843 and MAE of 67.50273, which is better than the multiple linear regression model. We can see that with limited information, predicting the end-price of an auction is quite difficult.

1. Predicting T+1 price with Static Model

We want to build a model that can predict the auction outcomes, that is the auction price at a future time (T+1) using information from the present (T) and the past (T-1, T-2,…,1). Assuming the estimated model parameters did not change from time T-1 to time T, at time T (present), the model can make a prediction about the future price at time T + 1.

A simple illustrate of the structure of the model would be:

T-1 T T+1

training

featuresT-1 targetT

predicting

featuresT T+1

In order to follow the dynamic model scheme of the forecasting model, we want to prepare a training dataset that contains features available at T-1, and target, that is the livebid price at T.

BY changing the predicting target, using the same static features and the same modeling methods as in the previous section. The results remains the same with a slightly smaller MAE.

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | Rsquared | MAE |
| Training | 210.1425 | 0.4695338 | 112.7726 |
| Testing | 232.015148 | 0.436801 | 110.296938 |

|  |  |  |  |
| --- | --- | --- | --- |
| k | RMSE | RSquared | MAE |
| 5 | 173.9282 | 0.6382034 | 85.05969 |
| 7 | 176.1739 | 0.6288147 | 87.21982 |
| 9 | 180.0484 | 0.6126760 | 89.90720 |

A close up of a map

Description automatically generated

1. Predicting T+1 price with Dynamic Model

To capture the process of bidding I created a livebid with a 1 day lag as a feature for the prediction. The dynamic features also includes the number of bidders along the auction, average bidding value, the standard deviation of the bidding value, the bid time difference and the bid value difference between each bid and the price velocity to capture the on-going features of an auction procedure. Adding this evolving information, we can create several dynamic models.

LM

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | Rsquared | MAE |
| Training | 210.1425 | 0.4695338 | 112.7726 |
| Testing | 96.08527 | 0.436801 | 67.37456 |

KNN

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | RSquared | MAE |
| Training | 173.9282 | 0.6382034 | 85.05969 |
| Testing | 181.56717 | 0.6288147 | 85.78306 |

XGBOOST

A close up of text on a black background

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | RSquared | MAE |
| Testing | 34.1539519 | 0.9766968 | 11.8855213 |

With 500 iterations, the XGboost linear predictor gives the lowest testing MAE.

LSTM Neural Network

My final approach for this project is Long Short-Term Memory (LSTM) neural network. LSTM is a specific type of recurrent neural network which is more adequate when it comes to modeling long-range dependencies. The dynamic models from previous section only use the information from the T-1. The memory is very short. Any value that is output in one-time step becomes input in the next, but unless that same value is output again, it is lost at the next tick. To solve this, we can form a neural network with a Long Short-Term Memory (LSTM) layer. The LSTM introduces a “memory cell” value that is passed down for multiple time steps and being sent in parallel with the activation output.

We can set a time window, a period of time to look back and predict the next target. The neural network I built has three layers, the first one is a LSTM layer, the second is a dense layer with 32 knot and the third layer is a dropout layer to prevent overfitting and the fourth layer is a dense layer for the output.

A screenshot of a cell phone

Description automatically generated

A close up of a map

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
|  | RMSE | RSquared | MAE |
| Testing | 223.4743 | NA | 112.7578 |

Just like in any other neural network model, we have to rescale the input data to the range of the activation function. The default activation function for LSTM is sigmoid function whose range is [-1, 1]. Since the input of the neural network has to be normalized, the training error was very small. To compare the LSTM performance with other models, I revert the predicted values to the original scale and the MAE is 112.7578.

Conclusions

In this project I have implement several different machine learning methods to the online auction dataset. Just like other real-world data, the online auction dataset is a combination of time series type of data and cross-sectional type of data. So when building the learning models, there are many things to consider.

Our baseline model using Multiple Linear Regression models to predict the end-price with static features did not perform satisfactorily with a testing MAE of 184.0715687. Adding the on-going information such as the current price of concurrent auctions or the number of bidders in concurrent auctions to the model makes the performance much more accurate, where the time series played an importance role. With these dynamic features adding to the model, the testing MAE can be reduced to 11.8855213. The LSTM Neural Network did not return the best result part of the reason might be the insufficient amount of data that I have used for training. The unequal spaced time stamp might influence the result as well.

Future Works

An ensemble of unsupervised clustering results and linear regression might have higher predictive capabilities, since the method of XGboost linear predictor also combines the tree classification models and the linear models.

My future works also includes the practical application of auction predicting models that helps the buyers making bidding decisions. Making bidding is further complicated since there existence of many auctions that offer the same or similar item simultaneously. Thus we might need to consider the competitive bidding behaviors as well.

Appendices

Final\_Project\_FyonaSun

Fyona Sun

2/10/2020

library(readr)  
auction=read\_csv('./online-auctions-dataset/auction.csv')

## Parsed with column specification:  
## cols(  
## auctionid = col\_double(),  
## bid = col\_double(),  
## bidtime = col\_double(),  
## bidder = col\_character(),  
## bidderrate = col\_double(),  
## openbid = col\_double(),  
## price = col\_double(),  
## item = col\_character(),  
## auction\_type = col\_character()  
## )

head(auction)

## # A tibble: 6 x 9  
## auctionid bid bidtime bidder bidderrate openbid price item auction\_type  
## <dbl> <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <chr> <chr>   
## 1 1638893549 175 2.23 schaden… 0 99 178. Carti… 3 day aucti…  
## 2 1638893549 100 2.60 chuik 0 99 178. Carti… 3 day aucti…  
## 3 1638893549 120 2.60 kiwisst… 2 99 178. Carti… 3 day aucti…  
## 4 1638893549 150 2.60 kiwisst… 2 99 178. Carti… 3 day aucti…  
## 5 1638893549 178. 2.91 eli.fli… 4 99 178. Carti… 3 day aucti…  
## 6 1639453840 1 0.356 bfalconb 2 1 355 Carti… 3 day aucti…

dim(auction)

## [1] 10681 9

sapply(auction, class)

## auctionid bid bidtime bidder bidderrate openbid   
## "numeric" "numeric" "numeric" "character" "numeric" "numeric"   
## price item auction\_type   
## "numeric" "character" "character"

The data set includes 10681 observations and 9 variables auctionid: unique identifier of an auction bid: the proxy bid placed by a bidder bidtime: the time in days that the bid was placed, from the start of the auction bidder: eBay username of the bidder bidderrate: eBay feedback rating of the bidder openbid: the opening bid set by the seller price: the closing price that the item sold for (equivalent to the second highest bid + an increment) item: auction item auction\_type

Understanding eBay auction policy It’s important to understand how eBay auction policy works before conducting further analysis. I found this explanation easy to understand.

The website helps to answer the following questions:

What’s the relationship between bids and bidtime? Poeple have to bid higher than current price, which is the second highest bid+increment. If someone places a bid much higher than current price, eBay will only use part of his/her bid(second highest bid+increment) as the new current price. People can bid as long as their bid is higher than current price. Therefore, as bidtime increases, we sometimes see bids that are lower than previous bids. People will be notified when they are outbid, so they may increase their max bids to remain the higher bidder.

What’s the relationship between bids and closing price? Closing price=second highest bid+increment or highest bid in some cases when second highest bid+increment is greater than the thrid highest bid. For example, if third highest bid is 8 and second highest bid is 10, you can bid 10.01 and if no one’s bid is higher than yours, you get the item by paying 10.01 instead of 10+increment(which is 10.5)

Person who placed the highest bid gets the item but how much he/she needs to pay in most cases depends on the second highest bid. ## Exploring the dataset

## function bid.incr(price)  
bid.incr <-   
 function(price)   
 { bidint <- c(0,0.99,4.99,24.99,99.99,249.99,499.99,999.99,2499.99,4999.99)  
 ## interval of prices  
 inc <- c(0.05,0.25,0.5,1,2.5,5,10,25,50,100)  
 ## increments of price for particular interval  
 inc[findInterval(price,bidint)]  
 }  
############################################################  
########## make livebids ###########  
############################################################  
  
## function recover.livebids(bids,bidder,openbid,winningbid)  
## This function reconstructs the sequence of livebids by   
## inputing the raw bids,bidder IDs and opening bid  
  
recover.livebids <-   
 function(bids,bidder,openbid,winningbid)  
 { nbid <- length(bids)  
 if (nbid==1) { live <- openbid  
 live  
 }  
 else {  
 if (nbid==2) {live <- numeric(nbid)  
 live[1] <- openbid  
 live[nbid] <- winningbid  
 live}  
  
 else { live <- numeric(nbid)  
 maxbid <- bids[1]  
 live[1] <- openbid  
 live[nbid] <- winningbid  
 currentprice <- bids[1]  
 maxbidder <- bidder[1]  
 for (i in 2:(nbid-1))   
 { newbid <- bids[i]  
 newbidder <- bidder[i]  
 if (newbid > maxbid) { if (!(identical(newbidder,maxbidder)))   
 { incr <- bid.incr(maxbid)  
 if (newbid < maxbid+incr) currentprice <- newbid  
 else currentprice <- maxbid+incr  
 maxbid <- newbid   
 maxbidder <- newbidder  
 }   
 else {maxbid <- newbid   
 maxbidder <- maxbidder  
 }  
 }  
 else {if (newbid < maxbid) { incr <- bid.incr(newbid)  
 if (newbid+incr>=maxbid) currentprice <- maxbid  
 else currentprice <- newbid+incr  
 maxbid <- maxbid  
 maxbidder <- maxbidder  
 }  
 else { if (!(identical(newbidder,maxbidder)))   
 {currentprice <- maxbid  
 maxbid <- maxbid  
 maxbidder <- maxbidder  
 }  
 else {currentprice <- currentprice  
 maxbid <- maxbid  
 maxbidder <- maxbidder  
 }   
 }  
 }  
 live[i] <- currentprice  
 }  
 live  
 }  
 }  
 }

auction=read\_csv('./online-auctions-dataset/auction.csv')

## Parsed with column specification:  
## cols(  
## auctionid = col\_double(),  
## bid = col\_double(),  
## bidtime = col\_double(),  
## bidder = col\_character(),  
## bidderrate = col\_double(),  
## openbid = col\_double(),  
## price = col\_double(),  
## item = col\_character(),  
## auction\_type = col\_character()  
## )

swarovski=read\_csv('./online-auctions-dataset/swarovski.csv')

## Parsed with column specification:  
## cols(  
## Seller = col\_double(),  
## Bidder = col\_double(),  
## Weight = col\_double(),  
## Bidder.Volume = col\_double(),  
## Seller.Volume = col\_double()  
## )

head(auction)

## # A tibble: 6 x 9  
## auctionid bid bidtime bidder bidderrate openbid price item auction\_type  
## <dbl> <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <chr> <chr>   
## 1 1638893549 175 2.23 schaden… 0 99 178. Carti… 3 day aucti…  
## 2 1638893549 100 2.60 chuik 0 99 178. Carti… 3 day aucti…  
## 3 1638893549 120 2.60 kiwisst… 2 99 178. Carti… 3 day aucti…  
## 4 1638893549 150 2.60 kiwisst… 2 99 178. Carti… 3 day aucti…  
## 5 1638893549 178. 2.91 eli.fli… 4 99 178. Carti… 3 day aucti…  
## 6 1639453840 1 0.356 bfalconb 2 1 355 Carti… 3 day aucti…

head(swarovski)

## # A tibble: 6 x 5  
## Seller Bidder Weight Bidder.Volume Seller.Volume  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 332874919 718577508 2 3 547  
## 2 594667804 399983466 5 6 183  
## 3 663070601 655828811 1 4 274  
## 4 309608641 599835541 3 8 3986  
## 5 201729374 693022555 1 2 4681  
## 6 332874919 622536435 2 4 547

dim(auction)

## [1] 10681 9

dim(swarovski)

## [1] 200 5

summary(auction)

## auctionid bid bidtime bidder   
## Min. :1.639e+09 Min. : 0.01 Min. :0.000567 Length:10681   
## 1st Qu.:3.015e+09 1st Qu.: 72.00 1st Qu.:1.949931 Class :character   
## Median :3.021e+09 Median : 140.00 Median :4.140833 Mode :character   
## Mean :4.136e+09 Mean : 207.59 Mean :3.979628   
## 3rd Qu.:8.212e+09 3rd Qu.: 210.00 3rd Qu.:6.448060   
## Max. :8.216e+09 Max. :5400.00 Max. :6.999990   
##   
## bidderrate openbid price item   
## Min. : -4.00 Min. : 0.01 Min. : 26.0 Length:10681   
## 1st Qu.: 1.00 1st Qu.: 1.00 1st Qu.: 186.5 Class :character   
## Median : 5.00 Median : 4.99 Median : 228.5 Mode :character   
## Mean : 31.94 Mean : 52.25 Mean : 335.0   
## 3rd Qu.: 21.00 3rd Qu.: 50.00 3rd Qu.: 255.0   
## Max. :3140.00 Max. :5000.00 Max. :5400.0   
## NA's :11   
## auction\_type   
## Length:10681   
## Class :character   
## Mode :character   
##   
##   
##   
##

summary(swarovski)

## Seller Bidder Weight Bidder.Volume   
## Min. : 24273 Min. : 221355 Min. : 1.000 Min. : 1.00   
## 1st Qu.: 94870114 1st Qu.: 80389757 1st Qu.: 1.000 1st Qu.: 4.00   
## Median :201729374 Median :191077522 Median : 2.000 Median : 6.00   
## Mean :198089751 Mean :276467688 Mean : 3.085 Mean : 9.26   
## 3rd Qu.:309608641 3rd Qu.:449791350 3rd Qu.: 3.000 3rd Qu.:11.00   
## Max. :663070601 Max. :723365710 Max. :56.000 Max. :94.00   
## Seller.Volume   
## Min. : 2.0   
## 1st Qu.: 382.5   
## Median :1791.0   
## Mean :1966.3   
## 3rd Qu.:3986.0   
## Max. :4681.0

levels(as.factor(auction$item))

## [1] "Cartier wristwatch" "Palm Pilot M515 PDA" "Xbox game console"

#"Cartier wristwatch" "Palm Pilot M515 PDA" "Xbox game console"   
levels(as.factor(auction$auction\_type))

## [1] "3 day auction" "5 day auction" "7 day auction"

#"3 day auction" "5 day auction" "7 day auction"

## data manipulation

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.0 ✓ dplyr 0.8.5  
## ✓ tibble 2.1.3 ✓ stringr 1.4.0  
## ✓ tidyr 1.0.2 ✓ forcats 0.5.0  
## ✓ purrr 0.3.3

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(zoo)

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

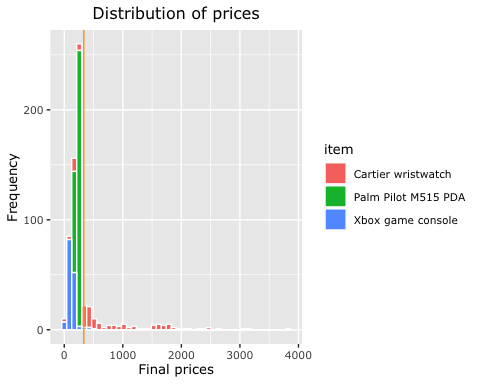
#change characteristic varibale to dummy variables  
auction$is\_3day<- ifelse(auction$auction\_type=='3 day auction',1,0)  
auction$is\_5day<- ifelse(auction$auction\_type=='5 day auction',1,0)  
auction$is\_7day<- ifelse(auction$auction\_type=='7 day auction',1,0)  
auction$is\_catier<- ifelse(auction$item=='Cartier wristwatch',1,0)  
auction$is\_palm<- ifelse(auction$item=='Palm Pilot M515 PDA',1,0)  
auction$is\_xbox<- ifelse(auction$item=='Xbox game console',1,0)  
#add a few statistical variables  
## winning bid, number of bids & average bid per auction  
u<-unique(auction$auctionid)  
winner<-rep(0,nrow(auction))  
livebid<- rep(0,nrow(auction))  
average<-rep(0,nrow(auction))  
sd<- rep(0,nrow(auction))  
num.bids<-rep(0,nrow(auction))  
  
for (i in u){  
 indexes<-which(auction$auctionid == i)  
 cur.average<-sum(auction$bid[indexes])/length(indexes)  
 cur.dev<- sd(auction$bid[indexes])  
 current<- recover.livebids(auction$bid[indexes],auction$bidder[indexes],auction$openbid[indexes],auction$price[indexes])  
 j<-0  
 for (k in indexes){  
 j<- j+1  
 winner[k]<-ifelse(auction$bid[k]==auction$price[k], 1 , 0)  
 num.bids[k]<-length(indexes)  
 average[k]<-cur.average  
 sd[k]<- cur.dev  
 livebid[k]<- current[j]  
 }  
  
}

auction$livebid.lag<- na.fill(auction$livebid.lag,0)

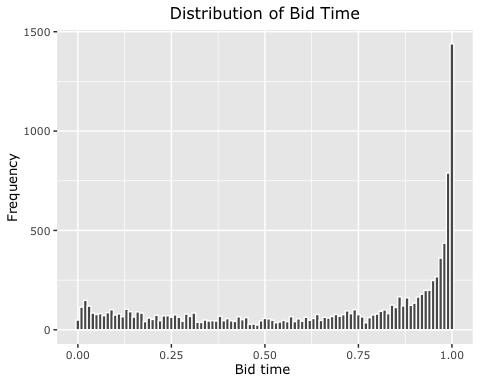
## Exploring the data

There are 627 item in the data set in total. Since the bidders do not know what others have bid and might be placing mulptiple bids after being notified that they were outbid, the last bid is not necessaily the one who get the item.

library(dplyr)  
#get rid of the outlier  
#1639672910  
auction<-auction[auction$auctionid!="1639672910",]  
  
closing\_price<- auction %>% select(auctionid,openbid,price,auction\_type,item)%>%  
 group\_by(auction$auctionid) %>%   
 summarise(openbid=max(openbid),price=max(price),auction\_type=max(auction\_type),item=max(item))  
# text theme for every plot in this project  
text\_theme <- theme(text = element\_text(size = 10,   
 family = "Verdana",   
 face = "plain"),  
 plot.title = element\_text(hjust = 0.5))  
  
# distribution of prices  
ggplot(closing\_price,   
 aes(x = price)) +   
 geom\_histogram(binwidth = 85,   
 aes(fill = item),   
 col = "white") +  
 geom\_vline(aes(xintercept = mean(auction$price)),   
 col = "orange",  
 size = 0.5) +  
 labs(title = "Distribution of prices",  
 x = "Final prices",  
 y = "Frequency") +  
 text\_theme

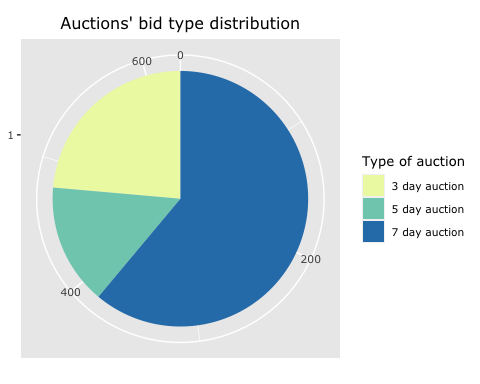


ggplot(auction,   
 aes(x = time.fraction)) +   
 geom\_histogram(binwidth = 0.01,  
 col = "white") +  
 labs(title = "Distribution of Bid Time",  
 x = "Bid time ",  
 y = "Frequency") +  
 text\_theme

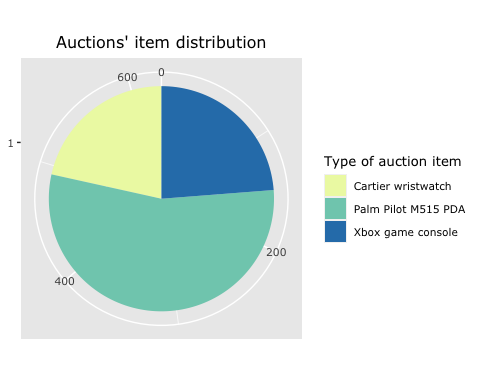


# distribution of auction types  
ggplot(closing\_price,   
 aes(x = factor(1),   
 fill = closing\_price$auction\_type)) +  
 geom\_bar(width = 3) +  
 ggtitle("Auctions' bid type distribution") +  
 coord\_polar(theta = "y") +  
 labs(fill= "Type of auction") +  
 scale\_fill\_brewer(palette = "YlGnBu") +  
 xlab(NULL) + ylab(NULL) +  
 text\_theme

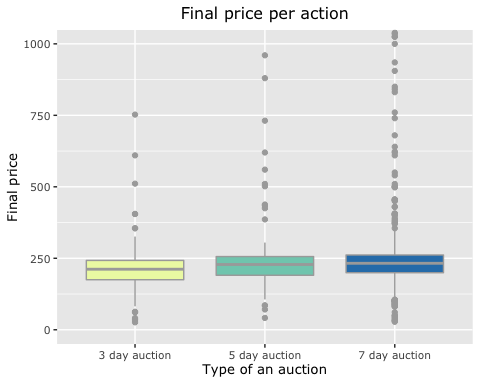
## Warning: Use of `closing\_price$auction\_type` is discouraged. Use `auction\_type`  
## instead.



# few ggplot complements   
bar\_auction\_type <- geom\_bar(aes(fill = auction$auction\_type),   
 stat = "count")  
  
ggplot(closing\_price,   
 aes(x = factor(1),   
 fill = item)) +  
 geom\_bar(width = 3) +  
 ggtitle("Auctions' item distribution") +  
 coord\_polar(theta = "y") +  
 labs(fill= "Type of auction item") +  
 scale\_fill\_brewer(palette = "YlGnBu") +  
 xlab(NULL) + ylab(NULL) +  
 text\_theme



# few ggplot complements   
bar\_auction\_type <- geom\_bar(aes(fill = auction$auction\_type),   
 stat = "count")  
  
# relationship between final price per auction and auction type  
ggplot(closing\_price,   
 aes(x = auction\_type, y = price)) +   
 geom\_boxplot(aes(fill = auction\_type),   
 color = "darkgrey") +  
 labs(title = "Final price per action",   
 x = "Type of an auction",   
 y = "Final price") +  
 guides(fill = F) +  
 scale\_fill\_brewer(palette = "YlGnBu") +   
 text\_theme +   
 coord\_cartesian(ylim = c(0, 1000)) # few outliers masked



## data manipulation

library(tidyverse)  
library(zoo)  
#change characteristic varibale to dummy variables  
auction$is\_3day<- ifelse(auction$auction\_type=='3 day auction',1,0)  
auction$is\_5day<- ifelse(auction$auction\_type=='5 day auction',1,0)  
auction$is\_7day<- ifelse(auction$auction\_type=='7 day auction',1,0)  
auction$is\_catier<- ifelse(auction$item=='Cartier wristwatch',1,0)  
auction$is\_palm<- ifelse(auction$item=='Palm Pilot M515 PDA',1,0)  
auction$is\_xbox<- ifelse(auction$item=='Xbox game console',1,0)  
#add a few statistical variables  
## winning bid, number of bids & average bid per auction  
u<-unique(auction$auctionid)  
winner<-rep(0,nrow(auction))  
livebid<- rep(0,nrow(auction))  
average<-rep(0,nrow(auction))  
sd<- rep(0,nrow(auction))  
num.bids<-rep(0,nrow(auction))  
  
for (i in u){  
 indexes<-which(auction$auctionid == i)  
 cur.average<-sum(auction$bid[indexes])/length(indexes)  
 cur.dev<- sd(auction$bid[indexes])  
 current<- recover.livebids(auction$bid[indexes],auction$bidder[indexes],auction$openbid[indexes],auction$price[indexes])  
 j<-0  
 for (k in indexes){  
 j<- j+1  
 winner[k]<-ifelse(auction$bid[k]==auction$price[k], 1 , 0)  
 num.bids[k]<-length(indexes)  
 average[k]<-cur.average  
 sd[k]<- cur.dev  
 livebid[k]<- current[j]  
 }  
  
}

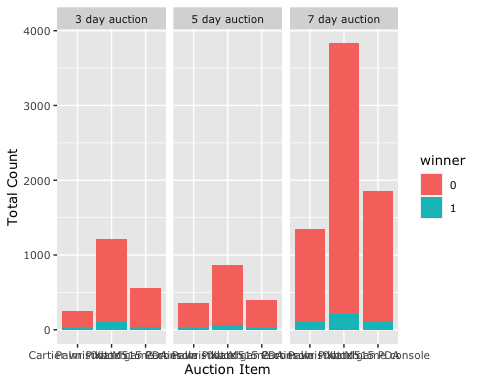
auction$winner<-winner  
auction$avg.bid<-average  
auction$sd.bid<- sd  
auction$num.bids<-num.bids  
auction$livebid<- livebid  
  
## length of auction (number of days)  
auction$length<-rep(0,nrow(auction))  
idx3<-which(auction$auction\_type == "3 day auction")  
auction$length[idx3]<-3  
idx5<-which(auction$auction\_type == "5 day auction")  
auction$length[idx5]<-5  
idx7<-which(auction$auction\_type == "7 day auction")  
auction$length[idx7]<-7  
  
## normalized bidding time (when bid was placed over auction length)  
auction$time.fraction<-auction$bidtime/auction$length  
  
## normalized bidding price (bid over winning price)  
auction$bid.fraction<-auction$bid/auction$price  
  
## bid distance from open bid  
auction<-mutate(auction,dist.open=(bid-openbid)/bid)  
  
## bid distance from average bid  
auction<-mutate(auction,dist.avg=(bid-avg.bid)/avg.bid)  
  
## add sequence of each bid  
auction$num\_current\_bids <- ave(auction$bidtime, auction$auctionid, FUN = seq\_along)  
  
## add time difference  
auction$time.diff <- ave(auction$bidtime, auction$auctionid, FUN = function(x) c(0,diff(x)))  
auction$time.diff[is.na(auction$time.diff)]<- 0   
  
## add bid difference  
auction$bid.diff <- ave(auction$bid, auction$auctionid, FUN = function(x) c(0,diff(x)))  
auction$bid.diff[is.na(auction$bid.diff)]<- 0   
  
## add bid velocity  
auction$velocity<- auction$bid.diff/auction$time.diff  
auction$velocity[is.infinite(auction$velocity)]<- 0   
auction$velocity[is.nan(auction$velocity)]<- 0   
  
##add lag  
auction$livebid.lag<- ave(auction$livebid, factor(auction$auctionid), FUN=function(x) c(NA,x[1:(length(x)-1)]))

auction$livebid.lag<- na.fill(auction$livebid.lag,0)

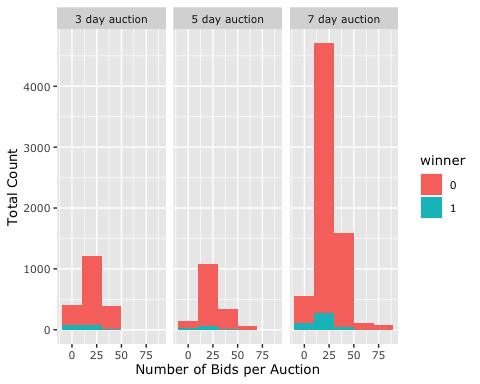
head(auction,50)

## # A tibble: 50 x 30  
## auctionid bid bidtime bidder bidderrate openbid price item auction\_type  
## <dbl> <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <chr> <chr>   
## 1 1.64e9 175 2.23 schad… 0 99 178. Cart… 3 day aucti…  
## 2 1.64e9 100 2.60 chuik 0 99 178. Cart… 3 day aucti…  
## 3 1.64e9 120 2.60 kiwis… 2 99 178. Cart… 3 day aucti…  
## 4 1.64e9 150 2.60 kiwis… 2 99 178. Cart… 3 day aucti…  
## 5 1.64e9 178. 2.91 eli.f… 4 99 178. Cart… 3 day aucti…  
## 6 1.64e9 1 0.356 bfalc… 2 1 355 Cart… 3 day aucti…  
## 7 1.64e9 1.25 0.485 sbord 1 1 355 Cart… 3 day aucti…  
## 8 1.64e9 1.5 0.493 bfalc… 2 1 355 Cart… 3 day aucti…  
## 9 1.64e9 25 0.495 sbord 1 1 355 Cart… 3 day aucti…  
## 10 1.64e9 2 0.511 bfalc… 2 1 355 Cart… 3 day aucti…  
## # … with 40 more rows, and 21 more variables: is\_3day <dbl>, is\_5day <dbl>,  
## # is\_7day <dbl>, is\_catier <dbl>, is\_palm <dbl>, is\_xbox <dbl>, winner <dbl>,  
## # avg.bid <dbl>, sd.bid <dbl>, num.bids <dbl>, livebid <dbl>, length <dbl>,  
## # time.fraction <dbl>, bid.fraction <dbl>, dist.open <dbl>, dist.avg <dbl>,  
## # num\_current\_bids <dbl>, time.diff <dbl>, bid.diff <dbl>, velocity <dbl>,  
## # livebid.lag <dbl>

auction$winner<-as.factor(auction$winner)  
######## PLOTS  
# text theme for every plot in this project  
text\_theme <- theme(text = element\_text(size = 10,   
 family = "Verdana",   
 face = "plain"),  
 plot.title = element\_text(hjust = 0.5))  
  
# distribution of bids over item and aution type  
ggplot(auction,aes(x =item, fill= winner))+  
 geom\_bar()+  
 facet\_wrap(~auction\_type)+  
 xlab("Auction Item")+ ylab("Total Count")+  
 text\_theme



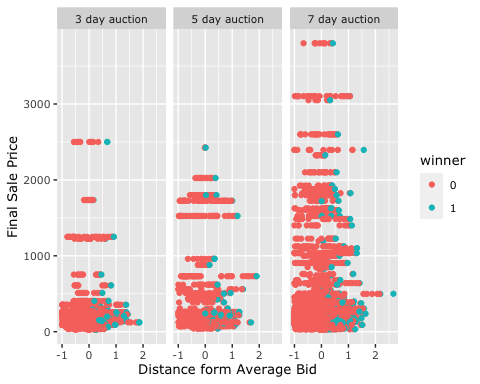
# check if no. of bids has predictive power  
# hypothesis - As number of bids per item increases chance of winning drops  
  
ggplot(auction,aes(x = num.bids, fill= winner))+  
 geom\_histogram(binwidth = 20)+  
 facet\_wrap(~auction\_type)+  
 xlab("Number of Bids per Auction")+ ylab("Total Count")+  
 text\_theme



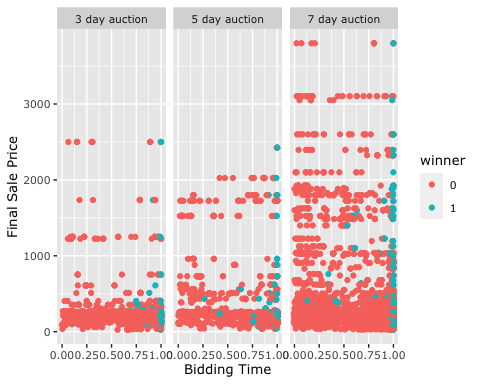
# check if distance from open bid has predictive power  
# hypothesis - Bids much higher than openning bid have higher chance of winning  
ggplot(auction)+  
 geom\_point(aes(x = dist.open,y=price, color= winner))+  
 facet\_wrap(~auction\_type)+  
 xlab("Distance form Openning Bid")+ ylab("Final Sale Price")+  
 text\_theme



# check if distance from average bid has predictive power  
# hypothesis - Bids below the average bid have zero chance of winning  
  
ggplot(auction)+  
 geom\_point(aes(x =dist.avg,y=price, color= winner))+  
 facet\_wrap(~auction\_type)+  
 xlab("Distance form Average Bid")+ ylab("Final Sale Price")+  
 text\_theme

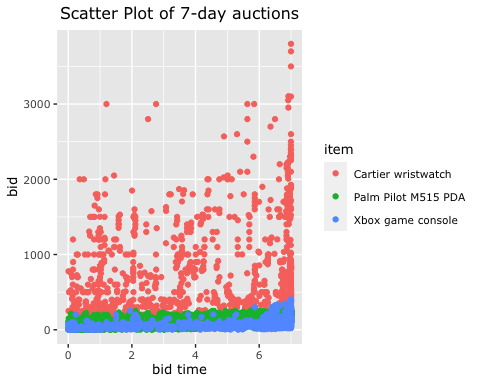


# check if time fraction has predictive power  
# hypothesis - Bids placed toward the end of the auction have higher chance of winning  
  
ggplot(auction)+  
 geom\_point(aes(x =time.fraction,y=price, color= winner))+  
 facet\_wrap(~auction\_type)+  
 xlab("Bidding Time")+ ylab("Final Sale Price")+  
 text\_theme

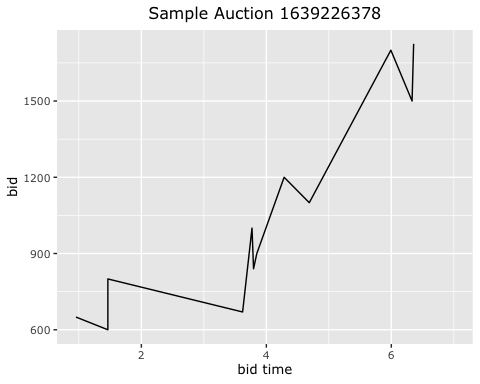


### modeling a single auction using Smoothing splines

#Example of a 7-day auction  
sevenday<- auction[which(auction$length==7),]  
ggplot(sevenday, aes(x=bidtime, y=bid, color= item)) +  
 geom\_point() +   
 scale\_fill\_brewer(palette = "YlGnBu") +   
 xlab("bid time") +  
 ggtitle("Scatter Plot of 7-day auctions")+  
 expand\_limits(x=7)+  
 text\_theme



sample3<- auction[which(auction$auctionid=='1639226378'),]  
ggplot(sample3, aes(x=bidtime, y=bid)) +  
 geom\_line() +   
 xlab("bid time") +  
 ggtitle("Sample Auction 1639226378")+  
 expand\_limits(x=7)+  
 text\_theme



## Model one single auction using non-linear polunomial regression

library(splines)  
library(gam)

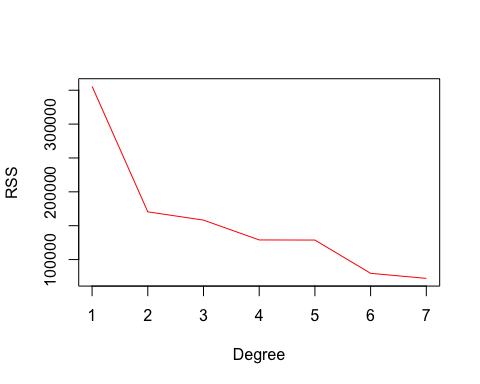
## Loading required package: foreach

##   
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':  
##   
## accumulate, when

## Loaded gam 1.16.1

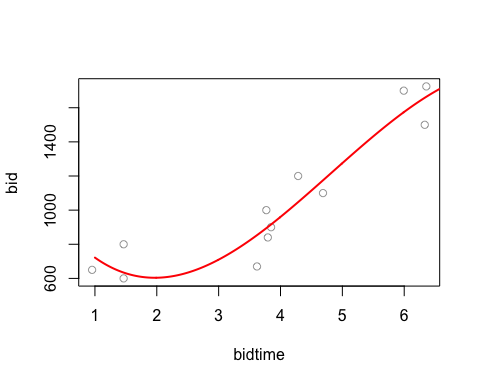
rss <- rep(0, 7)  
for (i in 1:7) {  
 fit <- lm(bid ~ poly(bidtime, i), data = sample3)  
 rss[i] <- sum(fit$residuals^2)  
}  
plot(1:7, rss, xlab = "Degree", ylab = "RSS", type = "l", col='red')



fit.lm <- lm(bid ~ poly(bidtime,3), data = sample3)  
summary(fit.lm)

##   
## Call:  
## lm(formula = bid ~ poly(bidtime, 3), data = sample3)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -183.30 -77.02 -25.10 110.84 165.86   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1057.1 40.6 26.039 5.08e-09 \*\*\*  
## poly(bidtime, 3)1 1178.1 140.6 8.377 3.13e-05 \*\*\*  
## poly(bidtime, 3)2 430.3 140.6 3.060 0.0156 \*   
## poly(bidtime, 3)3 -110.7 140.6 -0.787 0.4540   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 140.6 on 8 degrees of freedom  
## Multiple R-squared: 0.9093, Adjusted R-squared: 0.8752   
## F-statistic: 26.72 on 3 and 8 DF, p-value: 0.0001607

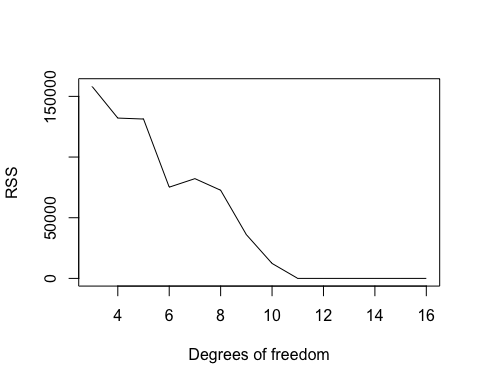
bidtime.new <- seq(1, 7, by = 0.01)  
pred <- predict(fit.lm, list(bidtime = bidtime.new))  
plot(bid ~ bidtime, data = sample3, col = "darkgrey")  
lines(bidtime.new,pred, col = "red", lwd = 2)



The RSS decreases as the degree of the polynomial decreases. It achieves its minimum at degree 7. However, it diverges as the degree of polynomial increases. Thus from the plot we choose the degree of 3.

## Soomthing splines and gam

rss <- rep(0, 16)  
for (i in 3:16) {  
 fit <- lm(bid ~ bs(bidtime, df = i), data = sample3)  
 rss[i] <- sum(fit$residuals^2)  
}  
plot(3:16, rss[-c(1, 2)], xlab = "Degrees of freedom", ylab = "RSS", type = "l")



fit.spline <- lm( bid~ bs(bidtime, knots = c(5, 14)), data = sample3)  
summary(fit.spline)

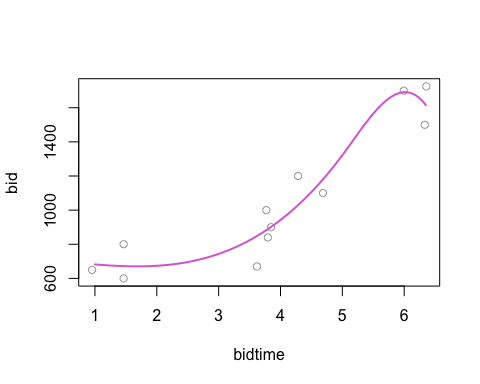
##   
## Call:  
## lm(formula = bid ~ bs(bidtime, knots = c(5, 14)), data = sample3)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -178.36 -74.05 -17.51 114.28 170.18   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 683.35 130.03 5.255 0.001179 \*\*   
## bs(bidtime, knots = c(5, 14))1 -41.55 506.34 -0.082 0.936893   
## bs(bidtime, knots = c(5, 14))2 -10.57 303.66 -0.035 0.973216   
## bs(bidtime, knots = c(5, 14))3 1150.59 312.44 3.683 0.007833 \*\*   
## bs(bidtime, knots = c(5, 14))4 928.39 161.69 5.742 0.000704 \*\*\*  
## bs(bidtime, knots = c(5, 14))5 NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 138.5 on 7 degrees of freedom  
## Multiple R-squared: 0.923, Adjusted R-squared: 0.879   
## F-statistic: 20.97 on 4 and 7 DF, p-value: 0.0005362

pred.spline <- predict(fit.spline, list(bidtime = bidtime.new))

## Warning in bs(bidtime, degree = 3L, knots = c(5, 14), Boundary.knots =  
## c(0.954039352, : some 'x' values beyond boundary knots may cause ill-conditioned  
## bases

## Warning in predict.lm(fit.spline, list(bidtime = bidtime.new)): prediction from  
## a rank-deficient fit may be misleading

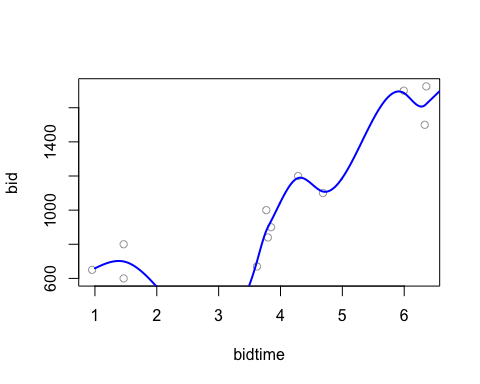
plot(bid ~ bidtime, data = sample3, col = "darkgrey")  
lines(bidtime.new, pred.spline, col = "orchid", lwd = 2)



bid<- sample3$bid  
bidtime<- sample3$bidtime  
fit.sspline <- smooth.spline(bid, bidtime,df=3)  
fit.gam<- gam(bid ~ s(bidtime,7))

## Warning in model.matrix.default(mt, mf, contrasts): non-list contrasts argument  
## ignored

pred.sspline<- predict(fit.sspline, bidtime.new)  
pred.gam <- predict(fit.gam, list(bidtime = bidtime.new))  
plot(bid ~ bidtime, data = sample3, col = "darkgrey")  
lines(bidtime.new,pred.sspline$y, col = "red", lwd = 2)  
lines(bidtime.new,pred.gam, col = "blue", lwd = 2)

 ## Add bidding pattern

#split by auction  
duration<-1  
sLive <- split(auction$livebid,auction$auctionid)  
sTime <- split(auction$time.fraction,auction$auctionid)  
#create variables with opening and closing bid added  
numauct<-length(sLive)  
sLiveEnds<-list(0);length(sLiveEnds)<-numauct  
sTimeEnds<-list(0); length(sTimeEnds)<-numauct  
sLiveStart<-list(0); length(sLiveStart)<-numauct  
sTimeStart<-list(0); length(sTimeStart)<-numauct  
sTimeEndsTemp<-list(0); length(sTimeStart)<-numauct  
sTimeStartTemp<-list(0); length(sTimeStartTemp)<-numauct  
for (j in 1:numauct){  
 sLiveEnds[[j]]<-c(min(sLive[[j]]),sLive[[j]],max(sLive[[j]]))  
 sTimeEnds[[j]]<-c(0,sTime[[j]],duration)  
 sLiveStart[[j]]<-c(min(sLive[[j]]),sLive[[j]])  
 sTimeStart[[j]]<-c(0,sTime[[j]])  
 sTimeEndsTemp[[j]]<-c(0.000001,sTime[[j]],duration)  
 sTimeStartTemp[[j]]<-c(0.000001,sTime[[j]])  
 }  
  
delta<- 0.01  
epsilon<-0.000001  
num.mod<-4 ##number of models  
w.x<-.5  
w.y<-.5  
  
  
########################  
##SSE for Diffent Models  
########################  
n<-length(sLive) ##number of auctions I'm looking at   
wSSE.range<-array(0,c(n,num.mod))   
wSSE.var<-array(0,c(n,num.mod))   
models.range<-array(0,n)  
models.var<-array(0,n)  
  
for (i in 1:n){  
  
## Exponential Model (Model 1)  
x<-seq(0,duration,by=.01)  
reg1<-lm(log(sLiveEnds[[i]])~ sTimeEnds[[i]])  
fit1y<-exp(as.vector(reg1$coefficients)[1])\*exp(as.vector(reg1$coefficients)[2]\*sTimeEnds[[i]])  
fit1x<-log(sLiveEnds[[i]]/exp(as.vector(reg1$coefficients)[1]))/as.vector(reg1$coefficients)[2]  
fit1.SSEy<- sum((fit1y-sLiveEnds[[i]])^2)  
fit1.SSEx<- sum((fit1x-sTimeEnds[[i]])^2)  
wSSE.range[i,1]<- (w.y\*fit1.SSEy)/diff(range(sLive[[i]]))^2 + (w.x\*fit1.SSEx)/diff(range(sTimeEnds[[i]]))^2  
wSSE.var[i,1]<- (w.y\*fit1.SSEy)/var(sLive[[i]]) + (w.x\*fit1.SSEx)/var(sTimeEnds[[i]])  
  
## Invere Exponential Model (Model 2)   
x<-seq(0,max(sLive[[i]]),by=.1)  
reg2<-lm(log(sTimeEndsTemp[[i]])~ sLiveEnds[[i]])  
fit2y<-log(sTimeEndsTemp[[i]]/exp(as.vector(reg2$coefficients)[1])) /as.vector(reg2$coefficients)[2]  
fit2x<-exp(as.vector(reg2$coefficients)[1]) \* exp(as.vector(reg2$coefficients)[2]\*sLiveEnds[[i]])  
fit2.SSEy<- sum((fit2y-sLiveEnds[[i]])^2)  
fit2.SSEx<- sum((fit2x-sTimeEnds[[i]])^2)  
wSSE.range[i,2]<- (w.y\*fit2.SSEy)/diff(range(sLive[[i]]))^2 + (w.x\*fit2.SSEx)/diff(range(sTimeEnds[[i]]))^2  
wSSE.var[i,2]<- (w.y\*fit2.SSEy)/var(sLive[[i]]) + (w.x\*fit2.SSEx)/var(sTimeEnds[[i]])  
  
## Logistic Growth (Model 3)   
x<-seq(0,duration,by=.01)  
reg3<- lm( log((max(sLiveStart[[i]])+delta)/sLiveStart[[i]]-1) ~ sTimeStart[[i]] )  
fit3y<- (max(sLiveEnds[[i]])+delta)/(1+exp(as.vector(reg3$coefficients)[1])\*exp(as.vector(reg3$coefficients)[2]\*sTimeEnds[[i]]))  
fit3x<- (log( (max(sLiveEnds[[i]])+delta)/sLiveEnds[[i]] - 1) - log (exp(as.vector(reg3$coefficients)[1])))/as.vector(reg3$coefficients)[2]  
fit3.SSEy<- sum((fit3y-sLiveEnds[[i]])^2)  
fit3.SSEx<- sum((fit3x-sTimeEnds[[i]])^2)  
wSSE.range[i,3]<- (w.y\*fit3.SSEy)/diff(range(sLive[[i]]))^2 + (w.x\*fit3.SSEx)/diff(range(sTimeEnds[[i]]))^2  
wSSE.var[i,3]<- (w.y\*fit3.SSEy)/var(sLive[[i]]) + (w.x\*fit3.SSEx)/var(sTimeEnds[[i]])  
  
## Inverse Logistic Growth (Model 4)  
x<-seq(0,max(sLive[[i]]),by=.1)  
reg4<- lm( log((duration+epsilon)/sTimeEndsTemp[[i]]-1) ~ sLiveEnds[[i]])  
fit4y<- (log(((duration+epsilon)/sTimeEndsTemp[[i]])-1) - as.vector(reg4$coefficients)[1]) /as.vector(reg4$coefficients)[2]  
fit4x<- (duration+epsilon)/(1+exp(as.vector(reg4$coefficients)[1])\*exp(as.vector(reg4$coefficients)[2]\*sLiveEnds[[i]]))  
fit4.SSEy<- sum((fit4y-sLiveEnds[[i]])^2)  
fit4.SSEx<- sum((fit4x-sTimeEnds[[i]])^2)  
wSSE.range[i,4]<- (w.y\*fit4.SSEy)/diff(range(sLive[[i]]))^2 + (w.x\*fit4.SSEx)/diff(range(sTimeEnds[[i]]))^2  
wSSE.var[i,4]<- (w.y\*fit4.SSEy)/var(sLive[[i]]) + (w.x\*fit4.SSEx)/var(sTimeEnds[[i]])  
  
}

for (k in 1:n){  
 wSSE.range[k,][wSSE.range[k,]=="NaN"]=(min(na.omit(wSSE.range[k,])+1))  
 wSSE.var[k,][wSSE.var[k,]=="NaN"]=(min(na.omit(wSSE.var[k,])+1))  
 models.range[k]<-(1:num.mod)[min(wSSE.range[k,])==wSSE.range[k,]]  
 models.var[k]<-(1:num.mod)[min(wSSE.var[k,])==wSSE.var[k,]]  
}

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

## Warning in min(na.omit(wSSE.var[k, ]) + 1): no non-missing arguments to min;  
## returning Inf

## Warning in models.range[k] <- (1:num.mod)[min(wSSE.range[k, ]) ==  
## wSSE.range[k, : number of items to replace is not a multiple of replacement  
## length

## Warning in models.var[k] <- (1:num.mod)[min(wSSE.var[k, ]) == wSSE.var[k, :  
## number of items to replace is not a multiple of replacement length

(1:n)[models.range!=models.var]

## [1] 13 14 17 42 55 65 71 84 143 145 152 NA 177 179 NA NA 204 NA 211  
## [20] NA 238 NA NA NA 306 NA NA NA 345 NA NA NA NA NA 414 NA NA NA  
## [39] NA 449 NA NA NA 487 514 527 557 559 587 615

table(models.range)

## models.range  
## 1 2 3 4   
## 301 5 86 212

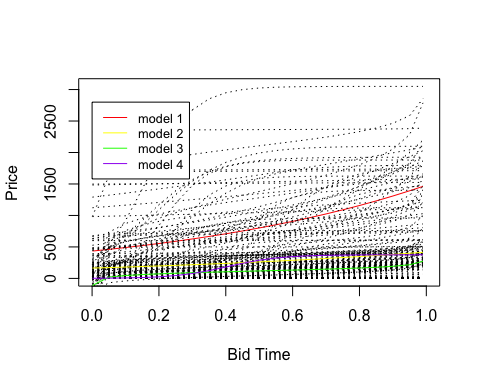
table(models.var)

## models.var  
## 1 2 3 4   
## 296 6 92 210

numauct<-length(sLive)  
x<-seq(0,duration,by=.1)  
x[1]<-min(x)+.001  
x[length(x)]<-max(x)-.01  
numplot<-length(x)  
ypred <- yfdpred7 <- ysdpred7 <- array(0,c(numauct,numplot))  
  
models<- models.range  
  
  
################################################################  
##Creating Fitted Data Based on Selected (by SSE criteria) Model  
################################################################  
## Matrix of data  
numauct<-length(sLive)  
x<-seq(0,duration,by=.05)  
x[1]<-min(x)+.001  
x[length(x)]<-max(x)-.01  
numplot<-length(x)  
ypred <- yfdpred7 <- ysdpred7 <- array(0,c(numauct,numplot))  
  
models7<- models<- models.range ###IMPT: THIS IS THE SSE CRITERIA THAT I AM USING (either models.range or models.var)   
  
for (i in 1:n){  
if (models[i]==1){  
reg1<-lm(log(sLiveEnds[[i]])~ sTimeEnds[[i]])  
ypred[i,]<-exp(as.vector(reg1$coefficients)[1])\*exp(as.vector(reg1$coefficients)[2]\*x)  
yfdpred7[i,]<-exp(as.vector(reg1$coefficients)[1])\*as.vector(reg1$coefficients)[2]\*exp(as.vector(reg1$coefficients)[2]\*x)  
ysdpred7[i,]<-exp(as.vector(reg1$coefficients)[1])\*(as.vector(reg1$coefficients)[2])^2\*exp(as.vector(reg1$coefficients)[2]\*x)  
}  
  
if (models[i]==2){  
reg2<-lm(log(sTimeEndsTemp[[i]])~ sLiveEnds[[i]])  
ypred[i,]<-log(x/exp(as.vector(reg2$coefficients)[1])) /as.vector(reg2$coefficients)[2]  
yfdpred7[i,]<-1/(as.vector(reg2$coefficients)[2]\*x)  
ysdpred7[i,]<- -1/(as.vector(reg2$coefficients)[2]\*x^2)  
}  
  
if (models[i]==3){  
reg3<- lm( log((max(sLiveStart[[i]])+delta)/sLiveStart[[i]]-1) ~ sTimeStart[[i]] )  
ypred[i,]<- (max(sLive[[i]])+delta)/(1+exp(as.vector(reg3$coefficients)[1])\*exp(as.vector(reg3$coefficients)[2]\*x))  
yfdpred7[i,]<- -(max(sLive[[i]])+delta)\*exp(as.vector(reg3$coefficients)[1])\*as.vector(reg3$coefficients)[2]\*exp(as.vector(reg3$coefficients)[2]\*x)/  
 (1+exp(as.vector(reg3$coefficients)[1])\*exp(as.vector(reg3$coefficients)[2]\*x))^2  
ysdpred7[i,]<- -(max(sLive[[i]])+delta)\*exp(as.vector(reg3$coefficients)[1])\*(as.vector(reg3$coefficients)[2])^2\*exp(as.vector(reg3$coefficients)[2]\*x)\*  
 (1-exp(as.vector(reg3$coefficients)[1])\*exp(as.vector(reg3$coefficients)[2]\*x))/  
 (1+exp(as.vector(reg3$coefficients)[1])\*exp(as.vector(reg3$coefficients)[2]\*x))^3  
}  
  
if (models[i]==4){  
reg4<- lm( log((duration+epsilon)/sTimeEndsTemp[[i]]-1) ~ sLiveEnds[[i]])  
ypred[i,]<- (log(((duration+epsilon)/x)-1) - as.vector(reg4$coefficients)[1]) /as.vector(reg4$coefficients)[2]  
yfdpred7[i,]<- -(duration+epsilon)/(as.vector(reg4$coefficients)[2]\*(x^2)\*(((duration+epsilon)/x)-1))  
ysdpred7[i,]<- ((duration+epsilon)\*((duration+epsilon)-2\*x))/(as.vector(reg4$coefficients)[2]\*(x^4)\*(((duration+epsilon)/x)-1)^2)  
}  
}

## Error in if (models[i] == 1) {: missing value where TRUE/FALSE needed

plot(x,ypred[1,],type="l",xlim=c(0,1),ylim=c(0,max(ypred)),xlab="Bid Time",ylab="Price",col='red')  
for (i in 2:numauct)  
lines(x,ypred[i,], lty=3)  
  
lines(x,ypred[86,], col='yellow')  
lines(x,ypred[61,], col='green')  
lines(x,ypred[9,], col='purple')  
  
legend(0, 2800, legend=c("model 1", "model 2", "model 3", "model 4"),  
 col=c("red", "yellow", 'green', 'purple'), lty=1, cex=0.8)



table(models)

## models  
## 1 2 3 4   
## 301 5 86 212

## Static model

library(dplyr)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

# Split data into training and testing  
set.seed(123)  
Train <- createDataPartition(u, p = 0.7, list = FALSE)  
Train.auction<- auction$auctionid %in% u[Train]  
training <- auction[Train.auction,]  
testing <- auction[!Train.auction,]

## static prediction

Static competition features are known at the outset of the auction and do not change during the auction; - opening price of concurrent auctions - duration of concurrent auctions - sellers’ feedback of concurrent auctions

library(ggplot2)  
library(caret)  
  
#######Static Model Predicting the End Price######  
set.seed(123)  
train\_control<- trainControl(method = 'cv',number=10,repeats = 3)

## Warning: `repeats` has no meaning for this resampling method.

#fit auction with linear model  
static.lm <- train(price~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day, data=training, trControl=train\_control, method="lm",na.action=na.exclude)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

static.lm

## Linear Regression   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6795, 6796, 6796, 6797, 6795, 6798, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 268.2168 0.5408452 132.4348  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

static.pred.lm<- predict(static.lm, testing)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

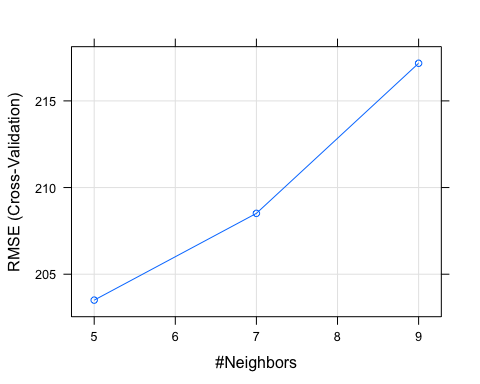
postResample(pred = static.pred.lm, obs = testing$livebid)

## RMSE Rsquared MAE   
## 303.4196504 0.3509475 184.0715687

##fit auction with knn model  
static.knn<- train(price ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day, data=training, trControl=train\_control, method="knn",na.action=na.exclude)  
static.knn

## k-Nearest Neighbors   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6797, 6796, 6796, 6795, 6795, 6796, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 203.5048 0.7333769 76.95662  
## 7 208.5114 0.7198521 82.10603  
## 9 217.1711 0.6960207 87.56857  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

plot(static.knn)



static.pred.knn<- predict(static.knn, testing)  
postResample(pred = static.pred.knn, obs = testing$livebid)

## RMSE Rsquared MAE   
## 325.888647 0.268136 181.203730

library(ggplot2)  
library(caret)  
  
#######Static Model######  
set.seed(123)  
  
train\_control<- trainControl(method = 'cv',number=10,repeats = 3)

## Warning: `repeats` has no meaning for this resampling method.

#fit auction with linear model  
static.lm <- train(livebid ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day, data=training, trControl=train\_control, method="lm",na.action=na.exclude)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

static.lm

## Linear Regression   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6796, 6796, 6796, 6796, 6795, 6796, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 210.1425 0.4695338 112.7726  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

static.pred.lm<- predict(static.lm, testing)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

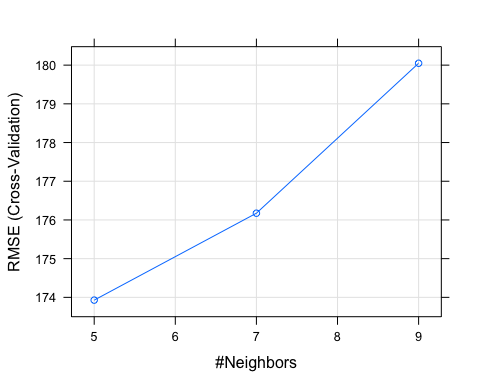
postResample(pred = static.pred.lm, obs = testing$livebid)

## RMSE Rsquared MAE   
## 232.015148 0.436801 110.296938

##fit auction with knn model  
static.knn<- train(livebid ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day, data=training, trControl=train\_control, method="knn",na.action=na.exclude)  
static.knn

## k-Nearest Neighbors   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6797, 6796, 6795, 6795, 6795, 6796, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 173.9282 0.6382034 85.05969  
## 7 176.1739 0.6288147 87.21982  
## 9 180.0484 0.6126760 89.90720  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

plot(static.knn)



static.pred.knn<- predict(static.knn, testing)  
postResample(pred = static.pred.knn, obs = testing$livebid)

## RMSE Rsquared MAE   
## 253.5584894 0.3333473 111.1309446

#######Static + Evolving info Model######  
#current price, time left, current number of bids, current number of bidders  
  
#fit auction with linear model  
se.lm <- train(livebid ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day+livebid.lag+num\_current\_bids+avg.bid+sd.bid+num.bids+time.fraction, data=training, trControl=train\_control, method="lm",na.action=na.exclude)

static.lm

## Linear Regression   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6796, 6796, 6796, 6796, 6795, 6796, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 210.1425 0.4695338 112.7726  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

se.pred.lm<- predict(se.lm, testing)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

postResample(pred = se.pred.lm, obs = testing$livebid)

## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length

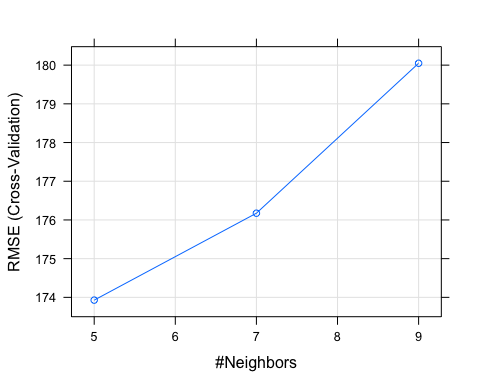
## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length

## RMSE Rsquared MAE   
## 95.92843 NA 67.50273

##fit auction with knn model  
se.knn<- train(livebid ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day+livebid.lag+num\_current\_bids+avg.bid+sd.bid+num.bids+time.fraction, data=training, trControl=train\_control, method="knn",na.action=na.exclude)  
static.knn

## k-Nearest Neighbors   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6797, 6796, 6795, 6795, 6795, 6796, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 173.9282 0.6382034 85.05969  
## 7 176.1739 0.6288147 87.21982  
## 9 180.0484 0.6126760 89.90720  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

plot(static.knn)



se.pred.knn<- predict(se.knn, testing)  
postResample(pred = se.pred.knn, obs = testing$livebid)

## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length  
  
## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length

## RMSE Rsquared MAE   
## 99.06266 NA 62.78169

#######Static + Evolving info + dynamic Model######  
#distance to   
  
#fit auction with linear model  
sed.lm <- train(livebid ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day+livebid.lag+num\_current\_bids+avg.bid+sd.bid+num.bids+time.fraction+bid.fraction+dist.open+dist.avg+time.diff+bid.diff+velocity, data=training, trControl=train\_control, method="lm",na.action=na.exclude)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading  
  
## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

static.lm

## Linear Regression   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6796, 6796, 6796, 6796, 6795, 6796, ...   
## Resampling results:  
##   
## RMSE Rsquared MAE   
## 210.1425 0.4695338 112.7726  
##   
## Tuning parameter 'intercept' was held constant at a value of TRUE

sed.pred.lm<- predict(sed.lm, testing)

## Warning in predict.lm(modelFit, newdata): prediction from a rank-deficient fit  
## may be misleading

postResample(pred = sed.pred.lm, obs = testing$livebid)

## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length

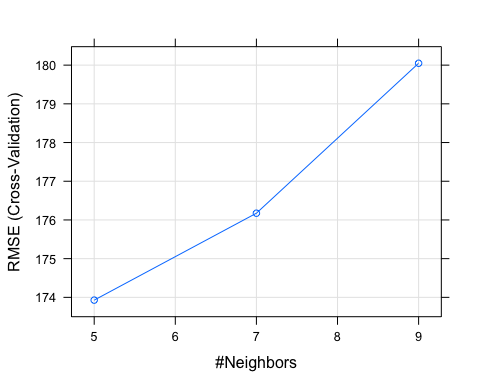
## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length

## RMSE Rsquared MAE   
## 96.08527 NA 67.37456

##fit auction with knn model  
sed.knn<- train(livebid ~ openbid+length+bidderrate+is\_catier+is\_palm+is\_xbox+is\_3day+is\_5day+is\_7day+livebid.lag+num\_current\_bids+avg.bid+sd.bid+num.bids+time.fraction+bid.fraction+dist.open+dist.avg+time.diff+bid.diff+velocity, data=training, trControl=train\_control, method="knn",na.action=na.exclude)  
static.knn

## k-Nearest Neighbors   
##   
## 7562 samples  
## 9 predictor  
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6797, 6796, 6795, 6795, 6795, 6796, ...   
## Resampling results across tuning parameters:  
##   
## k RMSE Rsquared MAE   
## 5 173.9282 0.6382034 85.05969  
## 7 176.1739 0.6288147 87.21982  
## 9 180.0484 0.6126760 89.90720  
##   
## RMSE was used to select the optimal model using the smallest value.  
## The final value used for the model was k = 5.

plot(static.knn)



sed.pred.knn<- predict(sed.knn, testing)  
postResample(pred = sed.pred.knn, obs = testing$livebid)

## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length  
  
## Warning in pred - obs: longer object length is not a multiple of shorter object  
## length

## RMSE Rsquared MAE   
## 181.56717 NA 85.78306

library(ggplot2)  
library(xgboost)

##   
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':  
##   
## slice

library(caret)  
  
#######XGboost Full model######  
set.seed(123)  
  
train\_x<- as.matrix(training %>% select(-which(sapply(.,is.character))) %>%select(-c(livebid,auctionid)))  
  
train\_y<- training$livebid  
  
test\_x<- as.matrix(testing %>% select(-which(sapply(.,is.character))) %>%select(-c(livebid,auctionid)))  
  
test\_y<- testing$livebid  
  
full.xg<- xgboost(data = train\_x,  
 label=train\_y,  
 booster = "gblinear",  
 objective = "reg:linear",  
 max.depth = 8,  
 nround = 500,  
 lambda = 0,  
 lambda\_bias = 0,  
 alpha = 0,   
 missing=NA,   
 verbose = 0)

## Error in xgb.DMatrix(data, label = label, missing = missing): 'data' has class 'character' and length 189050.  
## 'data' accepts either a numeric matrix or a single filename.

pred.test.xgboost <- predict(full.xg,test\_x, missing=NA)

## Error in predict(full.xg, test\_x, missing = NA): object 'full.xg' not found

rmse <- function(error)  
{  
 sqrt(mean(error^2))  
}  
   
# Function that returns Mean Absolute Error  
mae <- function(error)  
{  
 mean(abs(error))  
}  
  
error<- test\_y-pred.test.xgboost

## Error in eval(expr, envir, enclos): object 'pred.test.xgboost' not found

postResample(pred = pred.test.xgboost, obs = testing$livebid)

## Error in postResample(pred = pred.test.xgboost, obs = testing$livebid): object 'pred.test.xgboost' not found

## using LSTM to predicting the bid pattern

library(keras)  
  
lags <- function(x, k){  
   
 lagged = c(rep(NA, k), x[1:(length(x)-k)])  
 DF = as.data.frame(cbind(lagged, x))  
 colnames(DF) <- c( paste0('x-', k), 'x')  
 DF[is.na(DF)] <- 0  
 return(DF)  
 }  
  
lstm.train = lags(training$bid.diff, 1)  
lstm.test = lags(testing$bid.diff, 1)  
  
  
normalize <- function(train, test, feature\_range = c(0, 1)) {  
 x = train  
 fr\_min = feature\_range[1]  
 fr\_max = feature\_range[2]  
 std\_train = ((x - min(x) ) / (max(x) - min(x) ))  
 std\_test = ((test - min(x) ) / (max(x) - min(x) ))  
   
 scaled\_train = std\_train \*(fr\_max -fr\_min) + fr\_min  
 scaled\_test = std\_test \*(fr\_max -fr\_min) + fr\_min  
   
 return( list(scaled\_train = as.vector(scaled\_train), scaled\_test = as.vector(scaled\_test) ,scaler= c(min =min(x), max = max(x))) )  
   
 }  
  
   
Scaled = normalize(train, test, c(-1, 1))

## Error in min(x): invalid 'type' (closure) of argument

y\_train = Scaled$scaled\_train[, 2]

## Error in eval(expr, envir, enclos): object 'Scaled' not found

x\_train = Scaled$scaled\_train[, 1]

## Error in eval(expr, envir, enclos): object 'Scaled' not found

y\_test = Scaled$scaled\_test[, 2]

## Error in eval(expr, envir, enclos): object 'Scaled' not found

x\_test = Scaled$scaled\_test[, 1]

## Error in eval(expr, envir, enclos): object 'Scaled' not found

dim(x\_train) <- c(length(x\_train), 1, 1)

## Error in eval(expr, envir, enclos): object 'x\_train' not found

dim(x\_train)

## Error in eval(expr, envir, enclos): object 'x\_train' not found

batch\_size = 1  
units = 1  
# specify required arguments  
model <- keras\_model\_sequential() %>%   
 layer\_lstm(units=128,batch\_input\_shape = c(batch\_size, dim(x\_train)[2], dim(x\_train)[3]), stateful= TRUE)%>%  
 layer\_dropout(0.2) %>%   
 layer\_dense(units = 1)

## Error in normalize\_shape(batch\_input\_shape): object 'x\_train' not found

model %>% compile(  
 optimizer\_adam( lr= 0.02 , decay = 1e-6 ),  
 loss = "mse",  
 metrics = list("mean\_absolute\_error")  
)

## Error in eval(lhs, parent, parent): object 'model' not found

summary(model)

> summary(model)

Model: "sequential\_82"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

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

lstm\_71 (LSTM) (1, 128) 66560

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_22 (Dropout) (1, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_104 (Dense) (1, 1) 129

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

Total params: 66,689

Trainable params: 66,689

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

>

>

> history <- model %>% fit(

+ x\_train,y\_train,

+ epochs = 10,

+ batch\_size = 1

+ )

Train on 7474 samples

Epoch 1/10

7474/7474 [==============================] - 47s 6ms/sample - loss: 0.0070 - mean\_absolute\_error: 0.0409

Epoch 2/10

7474/7474 [==============================] - 21s 3ms/sample - loss: 0.0059 - mean\_absolute\_error: 0.0364

Epoch 3/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0060 - mean\_absolute\_error: 0.0360

Epoch 4/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0059 - mean\_absolute\_error: 0.0369

Epoch 5/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0059 - mean\_absolute\_error: 0.0361

Epoch 6/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0060 - mean\_absolute\_error: 0.0359

Epoch 7/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0056 - mean\_absolute\_error: 0.0355

Epoch 8/10

7474/7474 [==============================] - 21s 3ms/sample - loss: 0.0058 - mean\_absolute\_error: 0.0361

Epoch 9/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0060 - mean\_absolute\_error: 0.0359

Epoch 10/10

7474/7474 [==============================] - 22s 3ms/sample - loss: 0.0058 - mean\_absolute\_error: 0.0357

>

> dim(x\_test) <- c(length(x\_test), 1, 1)

> yhat = model %>% predict(X, batch\_size=1)

>

> postResample(pred = yhat, obs = y\_test)

RMSE Rsquared MAE

0.01653302 NA 0.01107428

>

history <- model %>% fit(  
 x\_train,y\_train,  
 epochs = 10,  
 batch\_size = 1

Model: "sequential\_88"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

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

lstm\_77 (LSTM) (1, 128) 66560

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_115 (Dense) (1, 32) 4128

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_28 (Dropout) (1, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_116 (Dense) (1, 1) 33

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

Total params: 70,721

Trainable params: 70,721

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

>

>

> history <- model %>% fit(

+ x\_train,y\_train,

+ epochs = 10,

+ batch\_size = 1

+ )

Train on 7557 samples

Epoch 1/10

7557/7557 [==============================] - 24s 3ms/sample - loss: 80823.0519 - mean\_absolute\_error: 141.0224

Epoch 2/10

7557/7557 [==============================] - 21s 3ms/sample - loss: 96979.0730 - mean\_absolute\_error: 158.2778

Epoch 3/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 104691.3947 - mean\_absolute\_error: 163.1122

Epoch 4/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 104151.2694 - mean\_absolute\_error: 161.5961

Epoch 5/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 103964.5418 - mean\_absolute\_error: 161.6179

Epoch 6/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 104444.3482 - mean\_absolute\_error: 161.5754

Epoch 7/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 104060.5758 - mean\_absolute\_error: 162.5403

Epoch 8/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 103962.0932 - mean\_absolute\_error: 161.4539

Epoch 9/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 104125.0715 - mean\_absolute\_error: 161.7306

Epoch 10/10

7557/7557 [==============================] - 22s 3ms/sample - loss: 104097.2588 - mean\_absolute\_error: 162.4093

>

> dim(x\_test) <- c(length(x\_test), 1, 1)

> yhat = model %>% predict(X, batch\_size=1)

>

> postResample(pred = yhat, obs = y\_test)

RMSE Rsquared MAE

223.4743 NA 112.7578