Card Kingdom Buy List Forecast

Wolf Of Tin Street

2/3/2021

Contents

Overview

The goal of this analysis is to predict what the buy list offer will be for any given card within a 28 day horizon period. A combination of both machine and deep learning methodologies will be utilized to take advantage of previously undocumented (or untracked) market indicators for this effect. In order to obtain external regressors as well as other attribute unique to each card, I will be taking advantage of not only my own scrapers and database storage, but the invaluable help and resources found over at **mtgjson.com**. If you have not investigated this resource yet, I'd highly advise you do so.

MTGJSON Roster Creation

I begin most, if not all, of my magic the gathering scripts by retreiving the most recent **AllPrintings** JSON from **https://mtgjson.com/downloads/all-files/#AllPrintings**. The most important reason for doing this is to ensure an accurate and current roster for all cards currently in circulation as well as early spoilers as they are released. For specifics on the fields themselves, they are incredibly well documented on the site, but I will point out the specific fields that I will be pulling out for later use.

rdate - The release date of every set **uuid** - MTGJSON's unique identifier for each card (foils would & do require slight adjustments) **scryfall id** - Another Identifier for scryfall to recognize **mcmid** - Magic Card Market's unique ID's for correspondance with EU markets **tcg_ID** - TCG Marketplace unique identifier for correspondance with NA markets **ckid**,**ckid_f** - Unique identifiers for Card Kingdom **card** - card name **set** - edition/set the card belongs to **abbr** - edition's three letter abbreviation **rarity** - The rarity of the card (Mythic, Rare, Uncommon, Common) **number** - Card's unique set number, and increasingly needed field given the recent tendency to print multiple versions of the same card in each set **types** - A unique card attribute in the game. Instants, Enchantments, Planeswalkers, etc. **manacost** - The cost required to play the card in game **colors** - There are 5 main colors of cards in mtg (Blue(U),Black(B),Green(G),Red(R),White(W)) with an auxilliary Brown (B) for colorless cards. **hasFoil** - Binary, does this card have a foil version. **hasNonFoil** - Binary, does this card have a nonfoil edition (commonly seen with Commander products) **hasAlternate** - Binary, does this card have an alternate version. This is usually used as a **fishing** field to try and locate premium versions of cards that might be otherwise missed by looking at the **hasFoil** field. **isPromo** - Binary, akin to the **hasAlternate** but tends to be looking more often than not for pre-release versions, or date stamped cards, that can only be obtained during special events surrounding set releases.

standard,**pioneer**,**modern**,**legacy**,**commander**,**pauper** - Binary fields for card legality in these specific, most popular, formats of play for the game.

isReserved - Binary, is this card part of the reserved list. Love it or hate it, this is a very important financial field. **edhrec_Rank** - Grouping of how many decks players self report to be including cards in. An esoteric and often unwieldly number to gauge consumer demand when access to financial metrics are not available. Given that most vendors do not have sales data outside their own, this can often be a powerful metric given the collective market continues to misuse &/or rely on it over stronger indicators. **legendary_commander** - Is the card specifically a legendary creature, thus allowing it to be used as a commander in EDH (EDH == commander)

Key - Arbitrary field I create for labeling. Combines the **card**,**set**,**rarity**,**hasFoil**,&**number** fields together as a human readable key. I could easily get away with simply **set**,**hasfoil**,&**number**, however, I, personally, enjoy the additional info provided.

```
## [1] "Error with set: Kaldheim Commander Promos"
```

Local Uploads

One of the many difficulties with Magic the Gathering e-commerce comparisons is that pretty much every site uses different set or edition names. By attempting to convert them all, whatever their version, back to the MTGJSON spelling gives us a unified game plan for normalizing for comparison. I have a local csv that I update to perform this task, however, **Koda** over at **https://github.com/kodabb/mtgban-website** has done a far superior job at scripting set conversions to MTGJSON, including as well the matching of the dreaded **promo edition**'s that (I believe ought to be) notoriously all over the place between vendors.

I also save to CSV and BQ (bigquery) the latest rendition of the MTGJSON Roster to ensure I always have some form of backup in the most up to date state as possible.

Datebase Pulls

I store my data in a BQ database in the cloud, as I am most familiar with that format and it suites my convenience of working off half a dozen droplets for mtg by allowing that information to be accessible at all times.

Now that I have an overall roster from MTGJSON and conversions elements in tibble format, I pull in historical market data from the different vendors.

Given that there are, currently, 56,000+ MTG cards in print, we have to use some kind of metric for creating a short pool of cards to review and forecast on.

In prior attempts, I would spend 8+ hours performing basic forecasting algorithms on single metrics (eg - Holts-Winter) on historical Buy List values only to forecast future buy list offers. This was not only inefficient and inaccurate in terms of accuracy, but time consumption as well, and it did not take into account market factors that we can reasonably expect to have an impact on the market place.

Increased complexity invariably results in longer processing periods, and given this current methodology, it will take 18 hours to perform apporximately 3000 forecasts given parallel processing with 8 cores.

Roster Addendum

The first task is to convert the **rdates** into a binary column to align with the time series format (each day in the historical or future window will get a 0 if not set released on that day, or a 1 if a set did release).

While the exact content of future sets cannot be known without insider or elicit information, we can include this element as an external regressor for gauging or explaining random price volatilites for certain cards if they occur in connection to a future set release date. Given our historical data used for modeling goes back over 240 days, we can reasonably expect to have at least 3 (if not more) set releases to help train/teach the model on this particular element.

Short List Creation

Since our end goal is predicting buy list measures, I elected to create an initial pool of any card that had a buy list offer over 50% vs their internal retail measure within the prior 8 months (240 days). As a short aside, Card Kingdom offers do not exceed 67% of what they will turn around and retail the card for, and conversely they tend not to drop below 15%. There are case exceptions to this, COVID lock down and individual high revenue generators (Sol Ring we're looking at you) but that is the general rule of thumb. So by pulling any card above this 50% threshold we are really narrowing down the higher performers.

A main goal is to try and locate and purchase cards at or below buy list, and it is not a realistic goal to expect, outside of collection purchases, to be able to locate cards that are selling on the market (generally a minimum of 20-100% higher price points) further below standing offers.

```
## # A tibble: 34,006 x 3
##    Key                                             number param
##    <chr>                                           <chr>  <chr>
##  1 Blue Elemental BlastUnlimited EditionC          50     8994
##  2 Lightning BoltFourth EditionC                   208    1825
##  3 Crimson KoboldsLegendsC                         139    3834
##  4 Lightning BoltRevised EditionC                  162    1479
##  5 SandstormArabian NightsC                        53     3237
##  6 Bojuka BogCommander 2011C                       267    47277
##  7 Wayfarer's BaubleDuel Decks: Venser vs. KothC 63     58128
##  8 Lava SpikeModern MastersC                       121    68382
##  9 Shadowborn ApostleMagic 2014C                   114    69196
## 10 SalvagePortal Second AgeC                       145    209
## # ... with 33,996 more rows
```

Now that we have a broader list of cards, we begin to further refine our pool. Percentages have a disproportionate effect on lower value cards, so I factor out any cards that average, during the prior 6 month period, less than a $3.50 **updated from 1.50** . Cards of this value, while certainly a potential opportunity for future analysis, are simply too expensive in terms of time and computation for the potential reward. Offers in this lower range also tend to be highly volatile, so they are removed from the pool of option.

Foils are also removed at this stage. Offers on foils are largely in name only. Most major vendors, Card Kingdom included, do not play 'fairly' with foils, in terms of grading. At best, upon submitting a NM (Near

Mint) foil, you have to expect to receive only 80% of that actual offer, if not worse. Not wanting to play this game, I have elected to ignore foils as well.

I will note, however, if one were to turn the axis of this forecasting onto predicting market values instead of buy list, I would highly advise the retention of foils as the open/broader market place has a different psychology and perception of foils as a whole and certainly price them in an inverse fashion to the large vendors.

Lastly, I remove Commons. As the name indicates, they are largely worthless. The commons that are of value are either extremely old or were printed in unique products like intro or duel decks. Locating supply on these has proven very difficult, and given their low price point from the start, are yet another example of where others may well find a niche I am currently avoiding for the sake of speed and performance.

Lastly, with this list of unique identifiers, I pull in their corresponding market metrics. These include: **BL** - Historical buy list offers **BL_QTY** - The amount of copies that were desired at that offer point on that particular day **MKT** - TCG Low Pricing. If we wanted to optimize our forecast for older/slower selling cards we'd likely want the market median as well. **Sellers** - The number of sellers (total) trying to sell the card on TCG's open market place **TCG_Rank** - The sales rate ranking of the cards via TCG's open market **CK_ADJ_Rank** - Card Kingdom's sales rate ranking.

It is important to note that both CK and TCG present their **best sellers** by revenue generation for themselves. Comically, however, they do provide enough information to apply a two step conversion via super complex math (divide and multiply) to arrive at an expected sales rate (even if Card Kingdom doesn't update their top two best sellers out of either laziness or to try and throw folks off…)

Forecasting Pre-processing

Due to, once again, the computationally expensive processes we are about to employ, my 16gb 8 core computer cannot handle doing more than 50 forecasts at a time without crashing under the strain. Therefore, I've had to break up our pool of cards into 50 card buckets to make the computations manageable.

I set a timer, mostly to satiate my own curiosity, but it also served to teach me early on how many forecasts/batches I could perform within a 24 hour span with the most recent market data available (settled at 2000 individual cards, or 40 batches of 50 individual cards).

Next, given the amount of time and energy (even if it's the terminators energy and not mine), I really wanted to maximize the analysis (though this could all have been done after the loop and might improve the speed ever so slightly).

week_combined_tbl: these allow me to break down which cards are predicted to simply go up in weeks 1,2,3, & 4. Sometimes pure consistency in direction is more telling than the granular forecast itself and I certainly want to keep a running tally and ranking of cards that are able to hit this base level of predicated growth.

expanded_all_forecasts: Allows me to store the entire forecasts results in case I do want to play with them later or if I have an idea. I eventually store this in a temporary daily table in BQ so I always have new toys to tinker with, as well as a copy of the most recent forecasts in case tomorrows script should fail or my upload of results to the **mtgban.com** site fail, they will be redirected to this table to ensure no gaps in total data provision.

boxplot_ranking_tbl: Creates a tier system, S-F & finally ignore, based off a boxplot visual in an effort to keep tier systems understandable. Important to note, the classification/rankings are based off expected growth with a base requirements of explained variance and an acceptable, but subjective, mae. A better classification would probably be one based off a gradation of both r^2 & mae, but folks like to see the money machine go "brrrrr" so I opted for this still of ranking mainly to draw attention to higher impact price changes, even if the confidence level might be lower than others.

```
## # A tibble: 50 x 5
##    dated_key                      Key                   Rarity    BL Date
##    <chr>                          <chr>                 <chr>  <dbl> <date>
##  1 Ajani VengeantShards of Alara~ Ajani VengeantShards ~ M       3.25 2020-11-14
##  2 Elspeth, Knight-ErrantShards ~ Elspeth, Knight-Erran~ M       6    2020-11-14
##  3 GodsireShards of AlaraM2020-1~ GodsireShards of Alar~ M       7    2020-11-14
##  4 Hellkite OverlordShards of Al~ Hellkite OverlordShar~ M       2.3  2020-11-14
##  5 Kresh the BloodbraidedShards ~ Kresh the Bloodbraide~ M       1.2  2020-11-14
##  6 Lich's MirrorShards of AlaraM~ Lich's MirrorShards o~ M       1    2020-11-14
##  7 Prince of ThrallsShards of Al~ Prince of ThrallsShar~ M       0.9  2020-11-14
##  8 Rafiq of the ManyShards of Al~ Rafiq of the ManyShar~ M       3.25 2020-11-14
##  9 Sarkhan VolShards of AlaraM20~ Sarkhan VolShards of ~ M       3.5  2020-11-14
## 10 Sedris, the Traitor KingShard~ Sedris, the Traitor K~ M       2.25 2020-11-14
## # ... with 40 more rows
```

Now that we have our empty tibbles and raw data, we can enter into the forecasting process. Due to the numerous transformations I wish to make before having our data in a palatable time series format, there are still some last minute alterations I need to do. Again, if I tried to apply these transformations outside the loop, my computer does not have the strength required, so yet another area for potential optimization.

Step one is selecting the batch we want to take in from the raw data. I do this by setting **a=1** & **b=50** outside the loop, and have additive measures within, immediately following batch selection, **a = a + 50** & **b = b + 50** with a conditional to end the loop should **b** ever become equal to or greater than **2000**, our 24 cut off period.

In limiting our selection, we also potential limit a very important field in our predictions, **rarity**. We need at least two different factors in the **rarity** column so the model is able to spot potential differences. In selecting only 50, it is extremely common to pull in only **rares** or **mythics** from one batch to the other. To account for this, I hack, a little bit, and append into our data the first **rare** or **mythic**, depending on which the batch requires, to provide at least on alternative example. This does mean that one **rare** or **mythic** will likely have a disproportionate effect on any such batch in every forecast. This is obviously an area where an improvement is very much desired, however, I'll take what I can get. Bucketing down to 50 is already sub-optimal, and this is just a further extension of that needed arrangement.

At this point, the card attributes from the MTGJSON roster are joined with the market data.

As yet another,but final, parameter, I require that every card, which ought to have 6 months of historical data have at least 100 days of non **NA** data. This immediately filters out cards from new sets. I do not wish to even begin to forecast for these cards until the market has at least 100 days to stabilize around these new cards. It also ensures that every card has a minimum amount of historical data when entering into the forecasting. It doesn't make much sense to try and use 10 days of market data to predict a future 30, the explained variance will be terrible and it will only serve to affect other potentially more accurate forecasts adversely.

Now, we do our final modifications in the **full_tbl**.

The first matter that needs to be addressed is that we have our full historical data set (or spreadsheet, if the verbiage makes it easier to visualize), but we need to add in date values for our desired forecast window (eg, how many days do we wish to forecast out) and fill in all other fields with **NA**'s. Once we have that, we can then address anomalies in our historical data set.

Important disclaimer, there have been some days where my scrapers have crashed.

A time series review requires that every day has a value, otherwise it will fail to run. Therefor, I need to adjust/account for these potential anomalies. I do this via the **pad_by_time** function - which, is absolutely amazing. It will automatically fill in, very similar to the forecast window we appended to the bottom of our data frame for the future, all missing dates with a row and the missing date, and fill in all other fields with **NA**'s. This saves me a great deal of time and effort trying to hunt down the specifics and really allows for automation to occur smoothly.

These blank fields, we will then use the **fill** function to fill in any **NA**'s from the pad by time with values from the prior day. Obviously, not ideal, however, needed for the algorithms and models to come to function. This does also have an unintended consequence, however. Now we've filled in all the values for our future columns with the most recent values that have occured. So if the market price today was $30, our forecast window, in this case 32 days, will all be filled in with this value. Obviously we do not know the future market values of the cards, so this will require some fixing.

Before we get to that particular fix, we also need to to adjust the numbers in our time series. A best selling ranking of 36,000 on Card Kingdom does not numerically compare well with a buy list offer of 5 dollars and desired quantity of 75. So we have to normalize our data - smoosh every value into a number between 0 & 1 to minimize the effect the larger numbers might have.

Going a little out of order than the code, but to circle back to that fill error, we need to lag our numerical data. We benefit, in forecasting for buy list values over market values, that buy list values are the slowest to move. They are standing cash offers after all. So what we're actually going to do, is lag these numerical fields. I've chosen the time periods of 32,42, & 56 days to lag our metrics. In doing this, I create a new column for every numerical column, and push it down 32,42, & 56 days. So the oldest dates in our data, will now have **NA**'s in those fields, and the metrics that occurred on day one, don't appear to start until day 32,42, & 56.

An analogy to what we're going to do would be if you take a book off your shelf. It has a sequel (*squints at Doors of Stone - Please Rothfuss help me out here*) that you just can't wait for. The story for the current book in your hand begins at Chapter 1, and ends at chapter 30. But we're greedy, and we really want to guess what will be in the next book. So we're going to change the game to try and predict what's in the next book. So we wipe clean all the pages in the book so there are no words. We then take all the writing that would have begun in Chapter 1 and overlay it onto where Chapter 5 previously was. In doing this, we'll end with 5 chapters at the end that not longer fit in our existing book. We then, essentially, pretend those 5 chapters are the start of the next awaited book. By doing this many, many times, we can try to actually predict the words that might go into the actual sequel when the book releases.

So why do we start at 32 days to lag? Because otherwise, we'd have filled in values from our most recent day of data being used as a potential future value for the data. We know this is incorrect from the off, but we still want to utilize our numerical data to the fullest extent we can. In forcing the values down an amount equal to our forecast window, we ensure the numerical values used are ALL real market values, and nothing I am filling in with placeholders. This is why we'll drop our original numerical columns at the very end, as they're using place holder metrics or containing NA's the will restrict the model from being able to run. We're also able to get away with this because we're forecasting for the buy list values, the last metric in the market, in my experience, to move in reframing a cards value into the future. Some may have legitimate arguments against that point, a strong one certainly being that the broader market sets prices far more than vendors, but I'll have those discussions elsewhere. For now, just be aware of that bias in the creation and usage of these metrics on my part.

We have now also multiplied the amount of information we have to predict on. Each numerical column was previously one column, but now we an additional three for each, which is gradually, more and more, pushing those fill'ed in current values out of the data set, giving us more realistic data, albeit lagged, to try and predict forward on. We further compound this by providing even more lag'ed periods wherein we take the moving 3,7,& 14 day averages of each numerical value and their lags to try and smooth out the changes over time. This averaging will allow the data to smooth out occasional spikes and anomalies and allow us to observe far more gradual and consistent movement. Given all these duplications of the numeric fields, we can easily drop the original numeric columns, although, obviously, not our original target BL column which we did not fill in on, to now have a nice, clean, data frame/spreadsheet that meets all the time series requirements. Last but not least, I also added fourier transformations at the 14,28,56, & 84 day marks in an effort to try and detect possible recurring frequencies in card values for the given time periods. Given that we're batching these time series together and not performing manual review of the pACf and ACF graphs, these fourier features would be an element I'd like to eventually circle back on and attempt to either utilize ML via **tune()** to locate the optimal periods or tailor in a similar automated fashion that is more in line with the data itself as opposed to my pre-processed periods of arbitrary choice.

```
##  [1] "rowid"                    "Key"
##  [3] "Rarity"                   "BL"
##  [5] "Date"                     "manaCost"
##  [7] "types"                    "colors"
##  [9] "printings"                "edhrecRank"
## [11] "set_release"              "Date_sin14_K1"
## [13] "Date_cos14_K1"            "Date_sin28_K1"
## [15] "Date_cos28_K1"            "Date_sin56_K1"
## [17] "Date_cos56_K1"            "Date_sin84_K1"
## [19] "Date_cos84_K1"            "BL_lag32"
## [21] "BL_lag42"                 "BL_lag56"
## [23] "BL_QTY_lag32"             "BL_QTY_lag42"
## [25] "BL_QTY_lag56"             "MKT_lag32"
## [27] "MKT_lag42"                "MKT_lag56"
## [29] "Sellers_lag32"            "Sellers_lag42"
## [31] "Sellers_lag56"            "TCG_Rank_lag32"
## [33] "TCG_Rank_lag42"           "TCG_Rank_lag56"
## [35] "CK_ADJ_Rank_lag32"        "CK_ADJ_Rank_lag42"
## [37] "CK_ADJ_Rank_lag56"        "BL_lag32_roll_3"
## [39] "BL_lag32_roll_7"          "BL_lag32_roll_14"
## [41] "BL_lag42_roll_3"          "BL_lag42_roll_7"
## [43] "BL_lag42_roll_14"         "BL_lag56_roll_3"
## [45] "BL_lag56_roll_7"          "BL_lag56_roll_14"
## [47] "BL_QTY_lag32_roll_3"      "BL_QTY_lag32_roll_7"
## [49] "BL_QTY_lag32_roll_14"     "BL_QTY_lag42_roll_3"
## [51] "BL_QTY_lag42_roll_7"      "BL_QTY_lag42_roll_14"
## [53] "BL_QTY_lag56_roll_3"      "BL_QTY_lag56_roll_7"
## [55] "BL_QTY_lag56_roll_14"     "MKT_lag32_roll_3"
## [57] "MKT_lag32_roll_7"         "MKT_lag32_roll_14"
## [59] "MKT_lag42_roll_3"         "MKT_lag42_roll_7"
## [61] "MKT_lag42_roll_14"        "MKT_lag56_roll_3"
## [63] "MKT_lag56_roll_7"         "MKT_lag56_roll_14"
## [65] "Sellers_lag32_roll_3"     "Sellers_lag32_roll_7"
## [67] "Sellers_lag32_roll_14"    "Sellers_lag42_roll_3"
## [69] "Sellers_lag42_roll_7"     "Sellers_lag42_roll_14"
## [71] "Sellers_lag56_roll_3"     "Sellers_lag56_roll_7"
## [73] "Sellers_lag56_roll_14"    "TCG_Rank_lag32_roll_3"
## [75] "TCG_Rank_lag32_roll_7"    "TCG_Rank_lag32_roll_14"
## [77] "TCG_Rank_lag42_roll_3"    "TCG_Rank_lag42_roll_7"
## [79] "TCG_Rank_lag42_roll_14"   "TCG_Rank_lag56_roll_3"
## [81] "TCG_Rank_lag56_roll_7"    "TCG_Rank_lag56_roll_14"
## [83] "CK_ADJ_Rank_lag32_roll_3"  "CK_ADJ_Rank_lag32_roll_7"
## [85] "CK_ADJ_Rank_lag32_roll_14" "CK_ADJ_Rank_lag42_roll_3"
## [87] "CK_ADJ_Rank_lag42_roll_7"  "CK_ADJ_Rank_lag42_roll_14"
## [89] "CK_ADJ_Rank_lag56_roll_3"  "CK_ADJ_Rank_lag56_roll_7"
## [91] "CK_ADJ_Rank_lag56_roll_14"
```

Splits

Now that we have our full, expanded data frame, we want to split it (70/30) into a training set, and a test set, in an effort to guage the accuracy of each forecasting methodology before ensembling, and later, applying to the future unknown to try and maximize our accuracy.

With our data split, we now have four data frames: 1) **future tbl** - this table consists exclusively of the future days. In our case this table contains our 28 days in the future we wish to predict for 2) **data__prepared__tbl** - our full 6 months of historical data 3) **training(splits)**- This consists of the oldest 70% of our data only 4) **testing(splits)** - This consists of the most recent 30% of our historical data

On all of our future models we will use the accuracy measurements from testing our forecasts from the **training(splits)** on our known **testing(splits)** and estimating they will retain those levels of accuracy 9or similar) into the future.

Next we need two **recipes()** created. We will be using 7 distinct models and algorithms, 5 Machine Learning based, and 2 Deep Learning, and each classification requires different instructions on how to deconstruct and treat each data frame for the algorithms underneath.

```
## # A tibble: 91 x 4
##    variable    type    role      source
##    <chr>       <chr>   <chr>     <chr>
##  1 rowid       numeric indicator original
##  2 Key         nominal predictor original
##  3 Rarity      nominal predictor original
##  4 Date        date    predictor original
##  5 manaCost    numeric predictor original
##  6 types       nominal predictor original
##  7 colors      nominal predictor original
##  8 printings   numeric predictor original
##  9 edhrecRank  numeric predictor original
## 10 set_release numeric predictor original
## # ... with 81 more rows
```

Forecasting

Models

5 distinct models will be used, and 8 overall.

1) Prophet_XGBoost
2) Ranger (Random Forests)
3) Multiplicative Adaptive Regression (MARs)

Originally, XGBoost and Prophet were run individually as well, but the combined form of Prophet_XGBoost consistently out performed either model in their own right, so for the sake of performance and brevity, they were eventually filtered out.

Also important to note, these initial three models are being utilized with their default parameters. Their metrics will be tuned later on, however MARs & Prophet XGBoost have a tendency to over fit after tuning. I retain the default and the tuned versions should they be top overall performers in the final ensemble, however. We will be calibrating, re-sampling, and refitting all non - deep learning models, and thus will rely on their general acurracy after those refinements.

A summary for these models, since we are using the **modeltime** package, may be found here **https://www.business-science.io/code-tools/2020/06/29/introducing-modeltime.html**

For deep learning we utilize two different forms of Deep AR 4) LSTM Deep Ar 5) ~~GRU Deep Ar~~ **Update** n_Beats now utilized given consistent LSTM over performance vs GRU.

Given the time constraints of re-sampling the Deep Ar epochs on the training data, we will not be performing that refinement on these two models. For reference, by resampling the AR models, computational time for a batch of 50 cards becomes 65 minutes. If we omit that re-sample, it is 20 minutes for a batch of 50 cards. The addition of this 40 minute refinement did not reveal a noticeable improvement in forecast accuracy, and was thus eventually omitted. The deep AR models actually tend to perform less well than more traditional algorithms, but I think that is likely due to the inadequate amount of historical data.

We then move on to tuning the Prophet_XGBoost, Ranger, & MARs models. For more detail on the parameters that require tuning, please see documentation for each model which can be found over at **https://www.tidymodels.org/find/parsnip/**. Also, we will be utilizing ~~3~~ **updated** to 7 K-folds for cross validation in tuning. This number could certainly be raised, but once again, time constraints play a major role. The additional k folds greatly increases the r-squared measurement which is a requirement later on, and thus the increased time dependencies was deemed worthwhile.

```
## [1] "Pre-tuning & Pre-resampling"
```

```
## # A tibble: 3 x 9
##   .model_id .model_desc                .type   mae  mape  mase smape  rmse   rsq
##       <int> <chr>                      <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1         2 RANGER                     Test  0.165  10.8 0.280  10.7 0.208 0.887
## 2         1 PROPHET W/ XGBOOST ERRORS Test  0.166  11.3 0.282  10.8 0.209 0.870
## 3         3 EARTH                      Test  0.213  13.6 0.361  13.1 0.274 0.827
```

Once all the raw and tuned models have been created and combined into one table, they are re-sampled in an attempt to further improve accuracy across a maximum of 5 slice elements with a 17 day spread between each for the models not utilizing deep learning.

Given that we did not resample or cross validate the deep learning models, the accuracy metrics will need to be pulled, aligned, and combined to create a singular table with the accuracy metrics of all algorithms presently cleanly for comparison.

The accuracy measurements, sorted by best **mae - mean absolute error** performance, will select the top 5 performing models for ensembling. The median instead of the mean, as the mean lead to consistent over fitting, and a manual weighting was not plausible for this automated procedure.

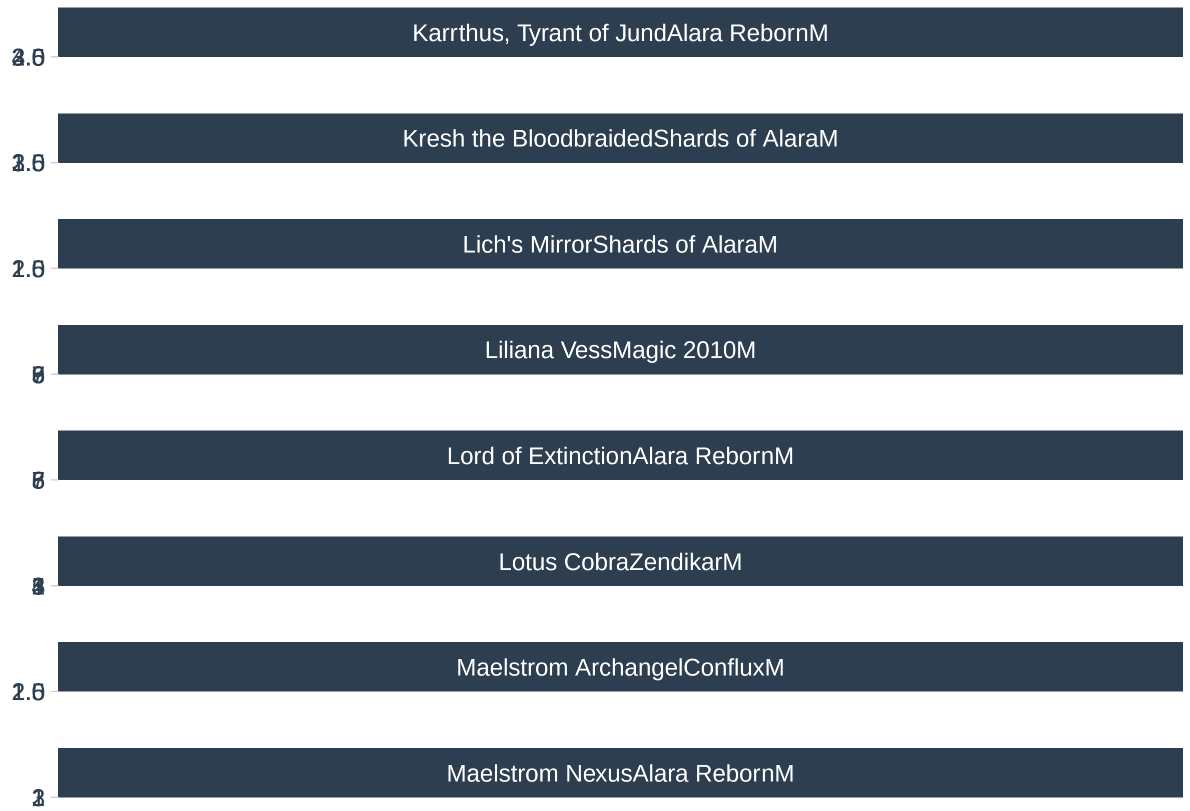
```
## -- Fitting Resamples --------------------------------------------
##
## 97.512 sec elapsed
```

```
## [1] "Accuracy metrics post tuning & resampling"
```

```
## # A tibble: 8 x 9
##   .model_id .model_desc              .type      mae  mape  mase smape  rmse   rsq
##       <int> <chr>                    <chr>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1         7 DEEPAR - LSTM            Test     0.142  9.79 0.241  9.28 0.192 0.891
## 2         8 DEEPAR - GRU             Test     0.155 10.3  0.263 10.6  0.209 0.887
## 3         5 RANGER                   Resampl~ 0.247 20.6  0.367 16.5  0.326 0.812
## 4         2 RANGER - Tuned           Resampl~ 0.250 20.6  0.371 16.6  0.328 0.811
## 5         1 PROPHET W/ XGB - Tuned   Resampl~ 0.251 20.4  0.373 16.5  0.334 0.774
## 6         4 PROPHET W/ XGBOOST ERR~ Resampl~ 0.261 21.1  0.389 17.3  0.345 0.771
## 7         3 EARTH - Tuned            Resampl~ 0.297 24.8  0.440 19.3  0.392 0.736
## 8         6 EARTH                    Resampl~ 0.308 22.1  0.457 21.6  0.399 0.705
```

With the new ensemble model created, the accuracy metrics for each individual card may be reviewed and stored for an accuracy range on the final forecast.

```
## # A tibble: 1 x 9
##   .model_id .model_desc                .type   mae  mape  mase smape  rmse   rsq
##       <int> <chr>                      <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1         1 ENSEMBLE (MEDIAN): 5 MODE~ Test  0.158  10.4 0.268  10.3 0.201 0.892
```



With our final model in hand, the ensemble model is retested against the **training(splits)** to gauge accuracy in a single data frame for every card, and is then refit to include the entirety of our historical data via the **data_prepared_tbl**.

Once the refitting has been accomplished, the forecast results are now used to populate the **week_combined_tbl**, **expanded_all_forecasts** & **boxplot_ranking_tbl** data frames that were previously empty.

```
## # A tibble: 6 x 8
##   Key       current_val   iqr range    sd max_forecast_va~ plus_minus Date      
##   <chr>           <dbl> <dbl> <dbl> <dbl>            <dbl>      <dbl> <date>    
## 1 Archange~        1.75  1.15  0.75 0.690             1.87      0.249 2021-04-16
## 2 Archange~        1.75  1.15  1.35 0.690             1.87      0.249 2021-04-16
## 3 Archange~        1.75  1.15  1.85 0.690             1.87      0.249 2021-04-16
## 4 Archange~        1.75  1.15  2.5  0.690             1.87      0.249 2021-04-16
## 5 Archange~        1.75  1.15  3    0.690             1.87      0.249 2021-04-16
## 6 Child of~        5.5   2.25  2    1.21              5.62      0.731 2021-04-15
```

```
## Time difference of 21.30274 mins
```

This process then repeats 40 times or until we exhaust all batches.

Forecast Analysis

As the most publicly visible results of this forecasting effort, seen on the **mtgban.com** newspaper page, stem from the **boxplot_ranking_tbl** data frame, I think it's processes and steps should be better broken down to best understand the resulting tier structure.

In the initial formation of the **boxplot_ranking_tbl** there were filtering steps that ought to be addressed.

1) Select the date, for every card in the forecast, the ensemble model predicted it would reach it's max buy list offer. This could be anywhere in the 28 day forecast period. The result is a single day plucked from the larger forecast demonstrating when it is expected to hit max offer.
2) We ensure that the +/- interval range is less than 80% of the cards own value. Once again, this serves to further flush out, more often than not, the lower value cards that are more susceptible to % volatility.
3) I require either r-squared of at least ~~35%~~ **update** 50% OR an rmse that is the top 20 of the batch of 50 cards. In as close to regular speak as I can put it, the model needs to be confidant that it can explain at least 50% of the variability surrounding the data. Another way of putting this is I'll never get published for this, but hey, this degree of explained variability is still able to consistently generate correct predictions and drive revenue. A lot of this forecasting does come down to human psychology on the open market place, so the low r-squared value, while certainly not ideal, does not make me panic. I could raise this criteria as well to be higher, but I've found that the output begins to lose it's value in predicting actual movement moving beyond this cutoff point. The rmse allowance is more to open the door to more volatile cards that may be experiencing volatility, but that the model can still reasonably predict the outcome for.

With this data frame, the final tier system is built. We just need to take our **boxplot_ranking_tbl** which has our max values, and combine it with a boxplot's measurements of the historical data for the card. These measurements include the interquartile range, the standard deviation,the median, and the outer limit value. This in combination with the known current value, the max forecasted value, and the +/- confidence range on the foresscast will determine the tiers.

S: Max forecast must be larger than the historical median offer. Max forecast must be larger than the historical median offer + standard deviation. Max forecast must be larger than the historical median offer + IQR. Max forecast must be 30% greater than the current offer. Max forecast minus the +/- value must still be greater than the outer limit of the boxplot. Max forecast must be at least $2 higher than current value. The Max forecast must be at least 5 days out from present date to allow for a buying opportunity.

A: Max forecast must be larger than the historical median offer. Max forecast must be larger than the historical median offer + standard deviation. Max forecast must be larger than the historical median offer + IQR. Max forecast must be 15% greater than the current offer. Max forecast minus the +/- value must still be greater than the outer limit of the boxplot. Max forecast must be at least $1 higher than current value. The Max forecast must be at least 5 days out from present date to allow for a buying opportunity.

B: Max forecast must be larger than the historical median offer. Max forecast must be larger than the historical median offer + standard deviation. Max forecast must be 15% greater than the current offer. Max forecast minus the +/- value must still be greater than the outer limit of the boxplot. Max forecast must be at least $0.5 higher than current value. The Max forecast must be at least 5 days out from present date to allow for a buying opportunity.

C: Max forecast must be larger than the historical median offer. Max forecast must be larger than the historical median offer + IQR. Max forecast must be 15% greater than the current offer. Max forecast minus the +/- value must still be greater than the outer limit of the boxplot. Max forecast must be at least $0.5 higher than current value. The Max forecast must be at least 5 days out from present date to allow for a buying opportunity.

D: Max forecast must be larger than the historical median offer. Max forecast must be 10% greater than the current offer. Max forecast minus the +/- value must still be greater than the outer limit of the boxplot.

Max forecast must be at least \$0.5 higher than current value. The Max forecast must be at least 3 days out from present date to allow for a buying opportunity.

E: Max forecast must be larger than the historical median offer. Max forecast must be larger than current value.

F: Max forecast must be larger than the historical median offer OR* larger than current value

Ignore: Meets none of the above specifications.

```
## # A tibble: 6 x 14
##   Key   name  Set   Rarity current_val   iqr    sd median_val outer_lim
##   <chr> <chr> <chr> <chr>        <dbl> <dbl> <dbl>      <dbl>     <dbl>
## 1 Arch~ Arch~ Port~ R             1.75   1.1   0.7       1.85       3
## 2 Eter~ Eter~ Zend~ M             4.20   1.2   0.7       3.5        4.5
## 3 Dark~ Dark~ Magi~ M             7.5    2     1.6       6.9        9
## 4 Jace~ Jace~ Magi~ M             2.75   0.5   0.5       2.05       3.2
## 5 Time~ Time~ Magi~ M            13.5    0.5   1.1      13         15
## 6 Chil~ Chil~ Conf~ M             5.5    2.2   1.2       4          6.9
## # ... with 5 more variables: max_forecast_value <dbl>, plus_minus <dbl>,
## #   Date <date>, Classification <chr>, Safety <chr>
```