Nash Domenichelli

T00562771

April 13, 2021

Comp 4980 Machine Learning

Analyzing StockX Yeezy Sales Data

Data Description

The data set I used is from a website called StockX which is a website that is primarily used for reselling sneakers. Some background on sneaker reselling is that there is a certain amount of hype behind a sneaker release and then a limited number of sneakers sold, so the resale prices are very high and there is a high demand because of the hype around them. The Adidas Yeezy brand is so popular because Kanye West is the designer, and they are one of the most common brands resold on StockX.

The data itself is only a few columns but with the data we are given, we can add more. The given columns are:

- Order Date: The date the show was ordered from StockX
- Brand: The dataset has Yeezy and Off-White but about 75% Yeezys in the dataset so its easier to stick with one
- Sneaker Name: For the most part this refers to only the colorway since we are looking at only Yeezys but there a couple different models in the dataset
- Sale Price: The price the sneakers were bought off the website. In USD
- Retail Price: The price the shoe was originally sold for buy the brand. In USD
- Release Date: The date the shoe was released by the brand
- Shoe Size: The size of the shoe that was sold
- Buyer Region: Where the person who bought the shoe was from in the USA

There were two additional columns added in a demo I found that I used as well:

- Time Released: The number of months the sale was made after retail release
- Profit: Sale Price – Retail Price

The exact data came from a StockX contest in 2019 where people submitted different findings and charts with the dataset. Here is the websites description of the data:

**The Data:**

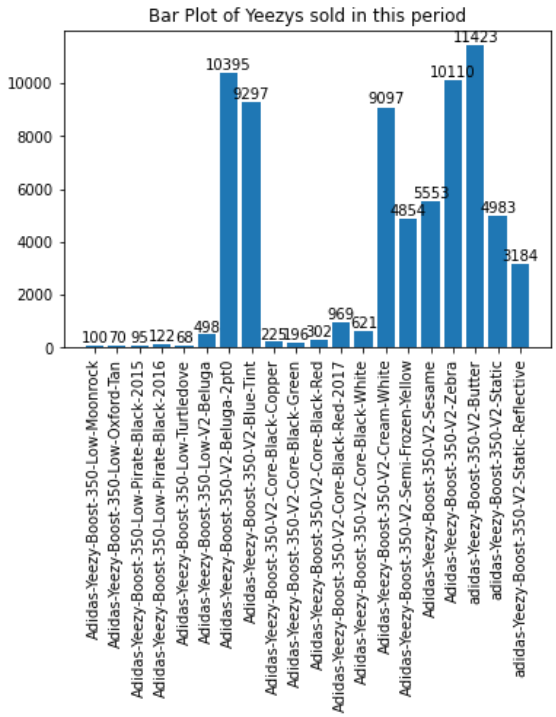
The data we're giving you consists of a random sample of all Off-White x Nike and Yeezy 350 sales from between 9/1/2017 (the month that Off-White first debuted "The Ten" collection) and the present. There are 99,956 total sales in the data set; 27,794 Off-White sales, and 72,162 Yeezy sales. The sample consists of U.S. sales only.

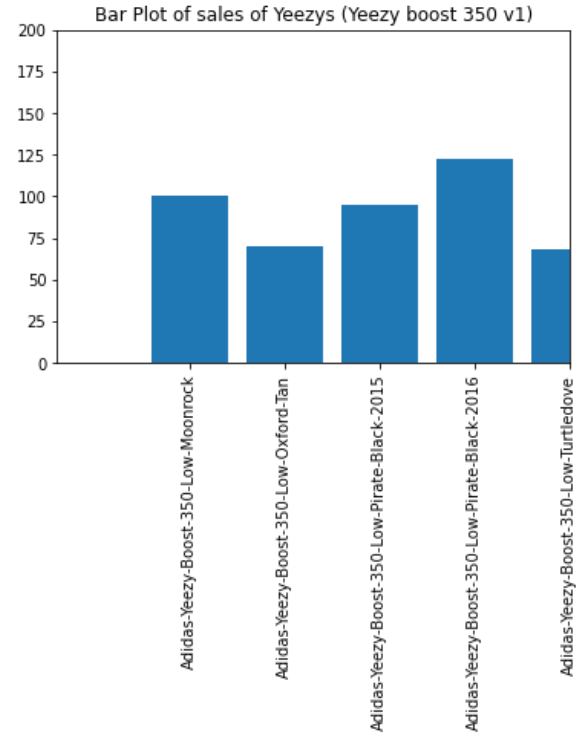
Clicking their link downloads an excel file of the data which I converted to csv format in excel. The data for the most was very usable but there were some issues where sometimes there was too much data to plot on a graph and if you tried to take samples of the data, it would not show all the points accurately.

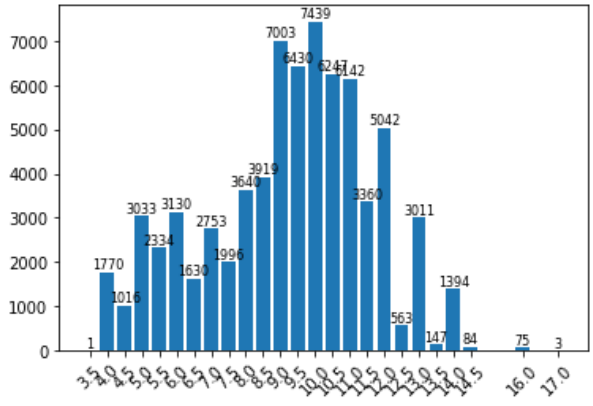
I wish the dataset had included sales of other types of Yeezys such as 500s and 700s and there were also data of 2020 sales because the market has changed a lot, but there is still more than enough data here to work with. There are also models of Yeezys owned by Nike instead of adidas that have seen ridiculous resale prices and data on those would have also been interesting.

Data Analysis

Just to get an idea of reselling culture, there are quite a few examples of general graphs to see the craze over sneaker reselling. This first graph shows the number of resales by the name of the sneaker. The second graph is just the first generation of Yeezys sold under Adidas and is a zoom on of the graph on the left.




Here is a graph of sales sorted by shoe size and it is interesting to see a natural distribution around size 10 with the half sizes having less sales than the whole numbers next to them for the most part.

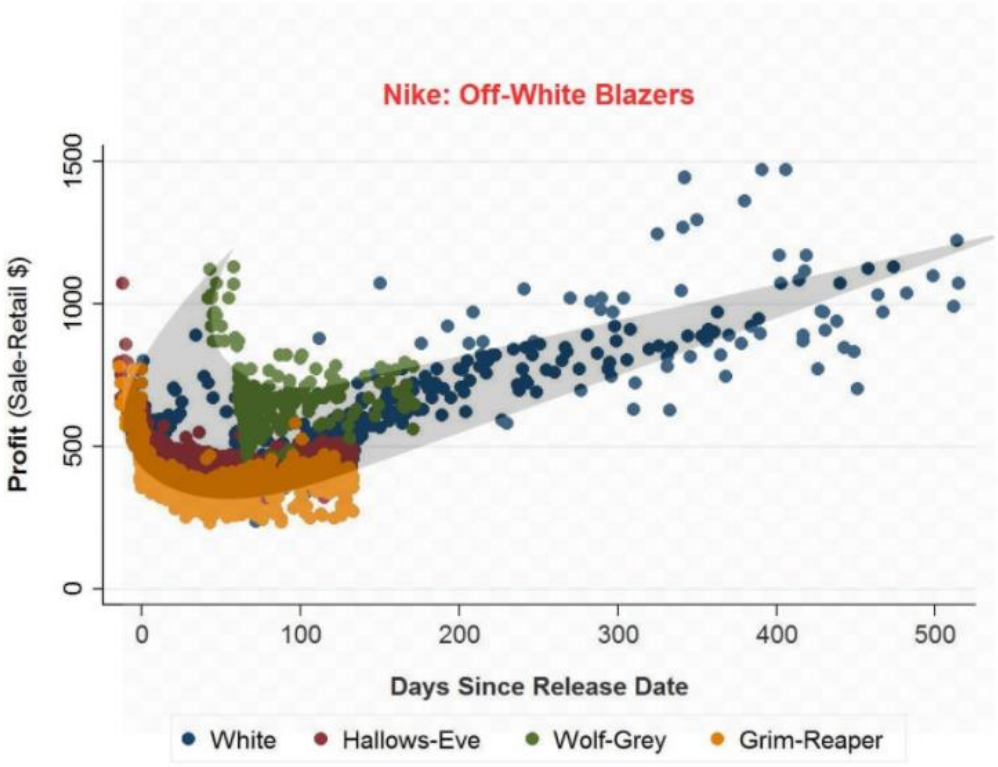
Another example of how crazy sneaker reselling is the mean profit is over $100 USD just for ordering them. StockX also only resales sneakers new in box, so they are more of a collection thing than buying the shoe to wear it.

```
yzy['Profit'].mean()

140.1586569108395
```

There are more interesting and in-depth analyses on the contest winners page using fancy graphics. One of the most unique ones using the same data set made Nike off white collab sales fit a Nike logo graphically.

**Aaron B. – The Blazer Swoosh Curve**

We've long noticed that resale prices for hyped sneakers, when plotted over time, often resemble the Nike Swoosh. This entry showed how Off-White Blazers follow this pattern closer than most:



Data Exploration

I did not find using PCA super useful because PCA is only super helpful when you have a lot of different data columns. For my research I did not use that many columns and had to add more instead of reducing data. I did the PCA from the slides and got a mean score of about 0.67 at best. I changed the value C, and I found 0.9 worked the best.

```
from sklearn.decomposition import PCA

pca = PCA(3)
pca.fit(X)
```

```
PCA(copy=True, iterated_power='auto', n_components=3, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

```
[9] pca.components_
```

```
array([[-4.70643813e-03,  5.76214196e-04,  6.71249952e-03,
         9.99966229e-01],
       [-3.04344051e-02,  2.72354301e-02,  9.99142051e-01,
        -6.86590348e-03],
       [ 1.10848760e-02, -9.99557833e-01,  2.75874591e-02,
         4.42963849e-04]])
```

```
[10] pca.explained_variance_ratio_
```

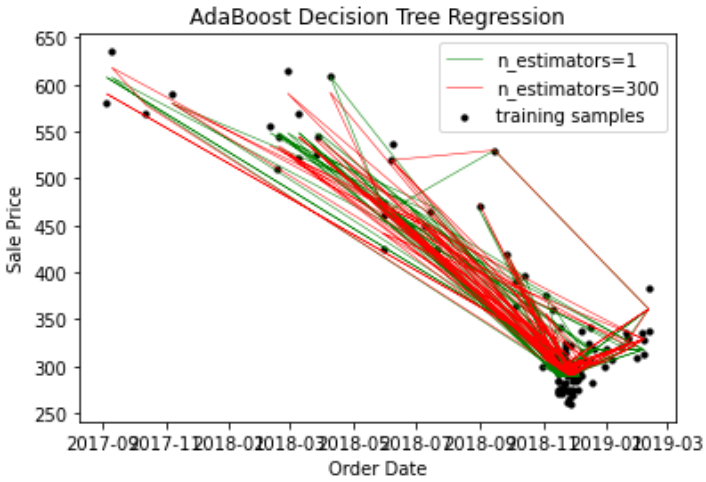
```
array([9.96626779e-01, 3.02655834e-03, 2.77695943e-04])
```

```
[11] Xt = pca.transform(X)
```

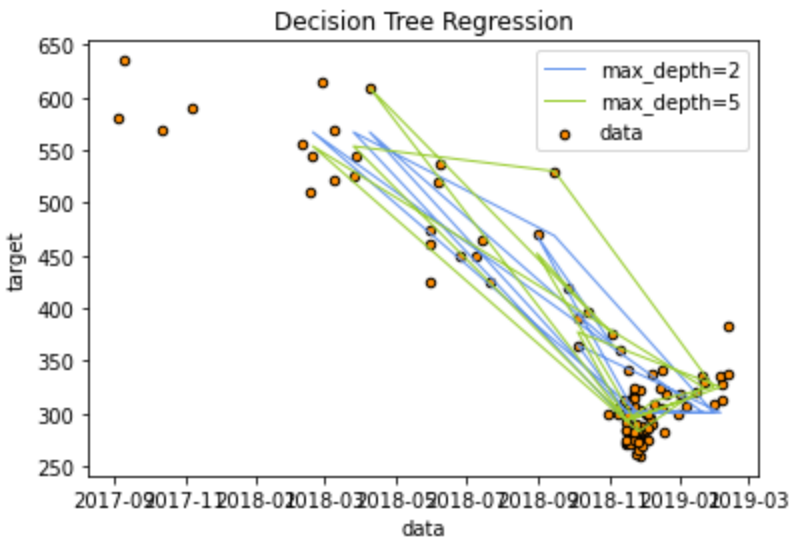
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import recall_score
from sklearn.model_selection import LeavePOut
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
#changing c to .9 from .1 got better scores
clf = SVC(kernel='linear', C=0.9)
cv = StratifiedKFold(n_splits = 8)

scores = cross_val_score(clf, Xt, y, cv=cv)

print(scores)
print(np.mean(scores))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_s
  % (min_groups, self.n_splits)), UserWarning)
[0.704 0.72  0.656 0.648 0.672 0.616 0.752 0.664]
0.6789999999999999
```

For decision trees I had more success, but the graphs can be hard to read because of the amount of data on them.




The black dots are the sales by order date with the price being on the y axis. This specific example was done on a sample of 100 sales of the Zebra colorway specifically. So, when they were released, they were reselling for much higher than their retail price at $220 but gradually went down over two years. There is also history with Adidas rereleasing sneakers and this graph shows that as well.




Here is another example of using decision trees but with different depths instead of AdaBoost. The lines are less crazy and show a more generalized example while the one with AdaBoost is a little overfit with lines going to every data point.

Looking up the history of rereleases, you can see why there are dips.

## Release History of Yeezy Zebra

- **Original Release:** February 25, 2017. The kicks scored a very very high resale value (around $1,500), so kudos to you if you were one of the people who flipped the kicks back then!
- **Restock #1:** June 24, 2017. Unfortunately to all the people that kept the kicks in hope their resale value will pick up, this release caused a huge drop in resale value.(Dropped to $565)
- **Restock #2:** November 9, 2018. After this restock, Yeezy Zebra scored the lowest resale value ever. ($286)
- **Yeezy Deadstock Day, Restock #3**: August 2, 2019. Before August 2, the resale value settled on an average ($365) that didn't change once the kicks dropped on Yeezy Day. That's probably due to the limited quantity of Zebras that were sold.
- **Now, Restock #4**: December 21, 2019. How low will it go?

This really depends on how much the asking prices would go. Will kids keep ruining the game and undersell their kicks and kill the hype altogether? Or will they finally value what they have and resell Yeezys for the prices they're worth?

Experimental Method

The first neural net I tried was a simple Multi Layer Perceptron Classification algorithm. I was getting very low accuracy when I was trying to predict sale prices accurately. With trying to predict the price to the dollar, I got 6% accuracy if I am understanding everything correctly which I think is surprisingly high for getting the exact dollar amount.

```
[27] yzyZebra = df[df['Sneaker Name'] == 'Adidas-Yeezy-Boost-350-V2-Zebra']
     yzyZebra2 = yzyZebra.copy()
     yzyZebra2 = yzyZebra.sample(1000)

     features = ['Shoe Size', 'Time Released']
     target = 'Sale Price'

     X = yzyZebra2[features]
     y = yzyZebra2[target]
```

```
[28] from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2)
```

```
[29] clf = nn.MLPClassifier(hidden_layer_sizes=(16,16,4), activation='relu', solver='adam', alpha=0.0001)
```

```
[30] cv = ms.StratifiedKFold()
     scores = ms.cross_val_score(clf, X, y, cv=cv, n_jobs=-1)

     print(np.mean(scores))

     /usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:667: UserWarning: The least p
       % (min_groups, self.n_splits)), UserWarning)
     0.066
```

```
[33] clf.fit(X_train, y_train)
     y_pred = clf.predict(X_test)
     print(accuracy_score(y_test,y_pred))

     0.045
```

The next logical step for predicting price was using a regression algorithm instead of classification algorithm. As expected, when not trying to pinpoint the exact price, it is easier to have a higher accuracy.

```
[34] import numpy as np
     import sklearn.neural_network as nn
     import matplotlib.pyplot as plt
     import sklearn.model_selection as ms
     from sklearn.metrics import r2_score
```

```
clf = nn.MLPRegressor(hidden_layer_sizes=(100,100), activation='relu',
                      solver='adam', alpha=0.0001, max_iter=1000)
```

```
[36] cv = ms.KFold()
     scores = ms.cross_val_score(clf, X, y, cv=cv, n_jobs=-1)

     print(np.mean(scores))

     0.8862946554499033
```

```
[38] clf.fit(X_train, y_train)
     y_pred = clf.predict(X_test)
     print(r2_score(y_test,y_pred))

     0.9275011704473686
```

Results and Analysis

A lot of the data was very understandable and there was nothing super shocking to me because I have a decent understanding of sneaker reselling markets specifically with Yeezys. I thought this would be a unique topic in the class and show insight to the crazy world of sneaker reselling without requiring work like image recognition I personally find interesting in machine learning. As I said in the beginning it would have also been interesting to get into 2020 data and data of other models but there was no data for it. The old data was also interesting for me personally because I got into it around the end of 2019, so it was interesting to research what happened before I was actively keeping up with it.

My original idea for this project was to use 2019 sales to predict 2020 sales and compare with how 2020 sales are using stockX's built in graphs. I could not find a way to easily put in a date especially when the date was in the future of the data, I had given it and there could be wrenches thrown in the consistency of the graphs such as rereleases as mentioned before or just sneaker releases having more stock in general so more people can buy at retail price.

Also 2020 data is going to be all over the place especially with the nature of luxury sneaker reselling. If people are only staying inside, they are not necessarily buying fancy shoes especially when I big part of the culture is to show them off. Recently some of the colorways are even dipping below retail for the first time ever and that could be due to anything mentioned above or other reasons like the brand being tied to Kanye West.

References


https://stockx.com/

https://stockx.com/news/the-2019-data-contest/

https://stockx.com/news/2019-data-contest-winner/

https://www.kaggle.com/sslp23/analyzing-yeezy-s-market

https://anbbot.com/yeezy-zebra-restock-2/

https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html#sphx-glr-auto-examples-tree-plot-tree-regression-py

https://scikit-learn.org/stable/auto_examples/ensemble/plot_adaboost_regression.html#sphx-glr-auto-examples-ensemble-plot-adaboost-regression-py

https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regression.html#sphx-glr-auto-examples-ensemble-plot-gradient-boosting-regression-py

Code Notebooks

Data Analysis:

https://colab.research.google.com/drive/1Bh3whHhVezcUQDje5XFF7hu25y3QE6dJ?usp=sharing

PCA:

https://colab.research.google.com/drive/1Zr7ZxiQ3JFUpUyVjMXvkBQSOdnEVO6n3?usp=sharing

Decision Trees:

https://colab.research.google.com/drive/1mzu5ptPFGoTvDgYbsojKIW1SMQcza96f?usp=sharing

Neural Nets:

https://colab.research.google.com/drive/1cJNE5KMWbvBwpVd7cE6YRYJ-RYXSk_S6?usp=sharing