Predicting Resale Value of Knives from a Texas Government Surplus Store

Using Machine Learning to Support an Ebay Store's Financial Success

Data Exploration and Modeling

Author: Dylan Dey

Overview

Texas State Surplus Store (https://www.tfc.texas.gov/divisions/supportserv/prog/statesurplus/)

What happens to all those items that get confiscated by the TSA? Some end up in a Texas store. (https://www.wfaa.com/article/news/local/what-happens-to-all-those-items-that-get-confiscated-by-the-tsa-some-end-up-in-a-texas-store/287-ba80dac3-d91a-4b28-952a-0aaf4f69ff95)

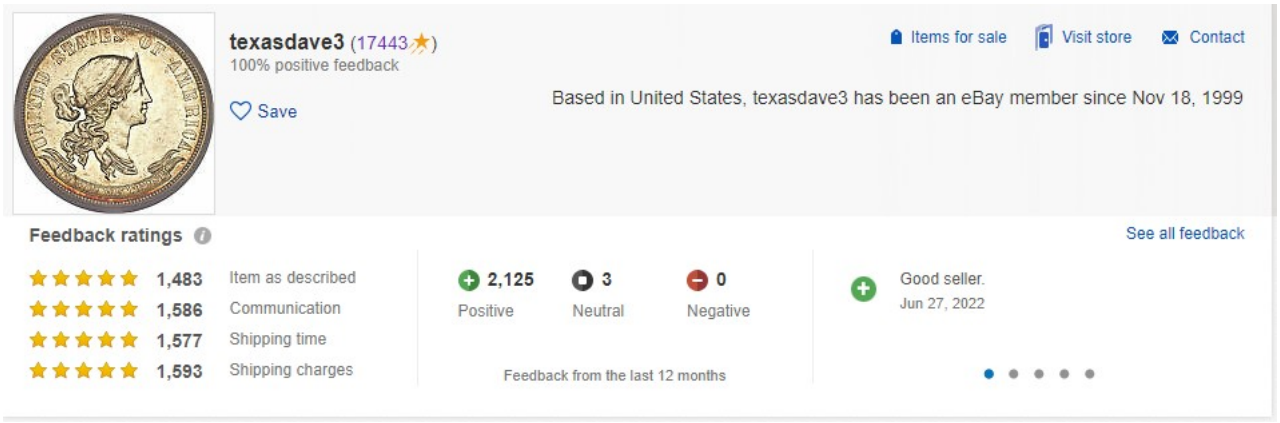
Texas Surplus Store PDF (https://www.tfc.texas.gov/divisions/supportserv/prog/statesurplus/State%20Surplus%20Brochure-one%20bar_rev%201-10-2022.pdf)

Everything that doesn't make it through Texas airports can be found at one Austin store (https://cbsaustin.com/news/local/everything-that-doesnt-make-it-through-texas-airports-can-be-found-at-one-austin-store)

> The Texas Facilities Commission collects left behind possessions, salvage, and surplus from Texas state agencies such as DPS, TXDOT, TCEQ, and Texas Parks & Wildlife. Examples of commonly available items include vehicles, furniture, office equipment and supplies, small electronics, and heavy equipment. The goal of this project is to create a predictive model in order to determine the resale value of knivse from the Texas State Surplus Store on eBay.

Business Problem

Family Ebay Store Front (https://www.ebay.com/str/texasdave3?mkcid=16&mkevt=1&mkrid=711-127632-2357-0&ssspo=ZW3G27tGR_m&sssrc=3418065&ssuid=&widget_ver=artemis&media=COPY)



Texas Dave's Knives (https://www.ebay.com/str/texasdave3/Knives/_i.html?store_cat=3393246519)

While taking online courses to transition careers during a difficult time of my life, I was also helping my family figure out how to improve their eBay success. I have been employed at their retail store in San Antonio for the past several months and have been contributing significantly to their online reselling business. I would help source newer, cheaper products from Austin to try and resell at the retail store in San Antonio or online to earn some money to support our family business. This is how I discovered the Texas State Surplus Store.

My family has been running a resale shop and selling on Ebay and other sites for years and lately the business has picked up. Consumer behavior is shifting: getting a deal on eBay, or Goodwill, or hitting up a vintage boutique shop to find a unique treasure is now brag worthy. Plus, people like the idea of sustainability - sending items to landfills is becoming very socially unacceptable – why not repurpose a used item? With the pandemic related disruption of “normal” business and supply chains and the economic uncertainty of these times there is definitely an upswing in interest in the resale market.

Online sales sites like Ebay offer a worldwide robust buyer base for just about every product regardless of condition. Ebay allows the reseller to find both bargain hunters for common items and enthusiasts searching for rare collectible items.

An Ebay business has some pain points, however. Selection of an item to sell is the main pain point. The item should be readily available in decent condition for the seller to purchase at a low price but not so widely available that the market is saturated with that item. Then there needs to be a demand for the item – it should be something collectible that with appeal to hobbyists that would pay premium prices for hard-to-get items. Alternatively, it would be something useful to a large number of people even in a used condition. The item should be small enough to be easily shipped. It should not be difficult to ship either—that is it should not have hazardous chemicals, batteries etc. that would add costs to the shipping. Additionally, Ebay has strict rules about authentication and certification in many item categories- so obvious “high value” items like jewelry or designer purses are so restricted that it is not feasible for the average Ebay seller to offer them.

The second pain point is buying at a cost low enough to make a profit. It is not enough to just buy low and sell at a higher price as expenses need to be considered. Ebay collects insertion fees and final value fees on all sales. The fees vary with seller level (rating) and some portions are a percent of final sale. I have been selling knives from the lower priced bins and the mean seller fee for my sales so far is about 13.5% of the sold price. So that is a cost to consider right up front.

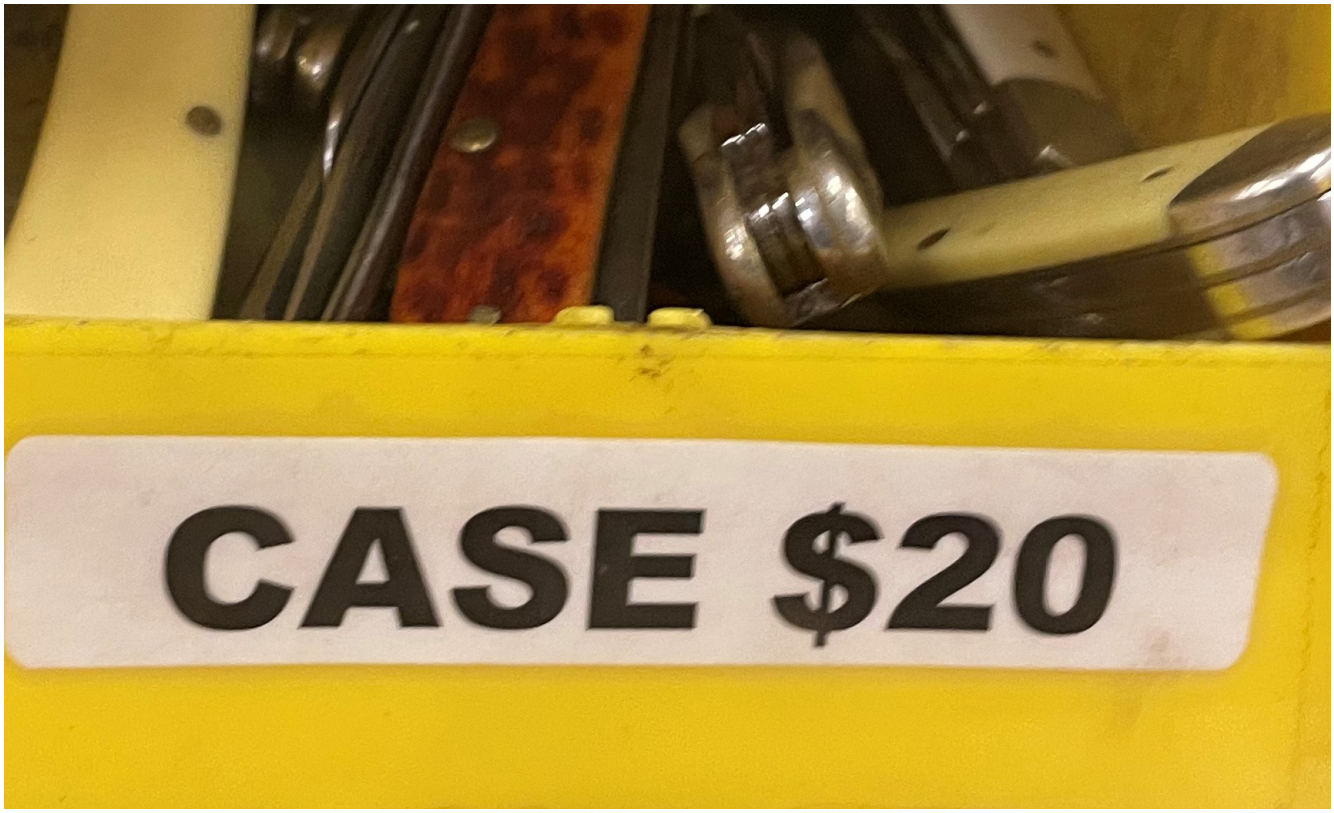
A third pain point is the cost of excess inventory. A seller can obtain quality items at a reasonable cost and then the inventory may sit with no sales, meaning the capital expended is sitting tied up in unwanted items. This inventory carry cost is a drain on profitability. This project is meant to help avoid purchasing the wrong items for resale.

This project recommends an item that would answer these concerns – pocket knives, These can be rare and collectible and also practical and useful. There are knife collector forums and subReddits, showing there is an interest among collectors. A look at eBay listings shows rare knives selling for thousands of dollars each. Knives are also a handy every day tool – and based on the number showing up in the Texas Surplus shop they are easy to lose and so need replacing often. This means there is a market for more common ones as well. The great thing about single blade, modern, factory manufactured pocketknives is that they all weigh roughly 0.5 lbs making them cheap to ship. For my modeling purposes, it is safe to assume a flat shipping rate of 4.95(US Dollars) including the cost of wholesale purchased padded envelopes. And there are no restrictions on mailing these items and they are not fragile so no special packaging is needed.

As already mentioned, I have been experimenting with low cost used knives for resale but have not risked a large capital investment in the higher end items. The goal of this project is to attempt to address the pain points to determine if a larger investment would pay off. Can I identify which knives are worth investing in so that I can turn a decent profit and hopefully avoid excess inventory? A data driven approach would help avoid costly mistakes from the "system" resellers currently employ, which seems to be mainly a gambler's approach. By managing resources upfront through a model, I can effectively increase my return on investment with messy data such as pictures and titles. The magic of Neural Networks!

There are eight buckets of presorted brand knives that I was interested in, specifically. These bins are behind glass, presorted, branded(and therefore have specific characteristics and logos for my model to identify), and priced higher. However, the staff has a very large amount of confiscated items flowing into the facility to list for resale, and when that happens they will not have time to preset them and they end up in huge buckets of unsorted knives for people to dig through. The brands will be priced the same, they are just no longer sorted and harder to find. This particular scenario is where a NN could really shine to help add more inventory to our Ebay website without risking more money or spending extra time than simply digging through the presorted bins everytime. Expanding the bins to pull inventory from will increase the chance of finding inventory worth reselling.

sorted bucket example



overflow example

Data Understanding

All of the data gathered from eBay's public API is limited to listed data posted in the past 90 days and doesn't include a "sold" price. Sold data is locked behind eBay's proprietary webapp, known as Terapeak. Data on this pay to play webapp has an option for sold data that goes back 2 years! Therefore, gaining access to this webapp and scraping all relevant pages proved to be very valuable and bypasses the limits of the free API. I used my reletavely new eBay seller's account to sign up for a free trial of terapeak and scraped useful data for sold, used knives of the 8 relevant brands. Information scraped includes Images, titles, price sold, shipping cost.

A majority of the data was scraped from eBays proprietary Terapeak webapp, as this data goes back 2 years as compared to the API listed data that only goes back 90 days. It is assumed a large enough amount of listed data should approximate sold data well enough to prove useful for this project.

The target feature for the model to predict is the total price (shipping included) that a knife should be listed on eBay. One model will be using titles and images in order to find potential listings that are undervalued and could be worth investing in. Another model will accept only images as input, as this is an input that can easily be obtained in person at the store. This model will use past sold data of knives on eBay in order to determine within an acceptable amount of error the price it will resale for on eBay (shipping included) using only an image.

Domain Understading: Cost Breakdown

padded envelopes: $0.50 per knife

flatrate shipping: $4.45 per knife

brand knife at surplus store: 15, 20, 30, or 45 dollars per knife

overhead expenses (gas, cleaning suplies, sharpening supplies, etc): $3.00

Ebay's comission, with 13% being a reasonable approximation

SCRUB/EXPLORE

In [1]:
```
import pandas as pd
import  json
import requests
import numpy as np
import re
# import preprocess_ddey117 as pp
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
import seaborn as sns
from collections import Counter
```

Define Functions

In [2]:

```
def apply_iqr_filter(df):

    price_Q1 = df['converted_price'].quantile(0.25)
    price_Q3 = df['converted_price'].quantile(0.75)
    price_iqr = price_Q3 - price_Q1

    profit_Q1 = df['profit'].quantile(0.25)
    profit_Q3 = df['profit'].quantile(0.75)
    profit_iqr = profit_Q3 - profit_Q1

    ROI_Q1 = df['ROI'].quantile(0.25)
    ROI_Q3 = df['ROI'].quantile(0.75)
    ROI_iqr = ROI_Q3 - ROI_Q1

    price_upper_limit = price_Q3 + (1.5 * price_iqr)
    price_lower_limit = price_Q1 - (1.5 * price_iqr)

    profit_upper_limit = profit_Q3 + (1.5 * profit_iqr)
    profit_lower_limit = profit_Q1 - (1.5 * profit_iqr)

    ROI_upper_limit = ROI_Q3 + (1.5 * ROI_iqr)
    ROI_lower_limit = ROI_Q1 - (1.5 * ROI_iqr)

#     print(f'Brand: {df.brand[0]}')
#     print(f'price upper limit: ${np.round(price_upper_limit,2)}')
#     print(f'price lower limit: ${np.round(price_lower_limit,2)}')
#     print('-----------------------------------')
#     print(f'profit upper limit: ${np.round(profit_upper_limit,2)}')
#     print(f'profit lower limit: ${np.round(profit_lower_limit,2)}')
#     print('-----------------------------------')
#     print(f'ROI upper limit: {np.round(ROI_upper_limit,2)}%')
#     print(f'ROI lower limit: {np.round(ROI_lower_limit,2)}%')
#     print('-----------------------------------')


    new_df = df[(df['converted_price'] <= price_upper_limit) &
                (df['converted_price'] >= price_lower_limit) &
                (df['profit'] <= profit_upper_limit) &
                (df['ROI'] <= ROI_upper_limit) &
                (df['profit'] <= profit_upper_limit) &
                (df['ROI'] >= ROI_lower_limit)]

    return new_df
#download jpg urls from dataFrame
def download(row):
    filename = os.path.join(root_folder, str(row.name) + im_extension)

# create folder if it doesn't exist
    os.makedirs(os.path.dirname(filename), exist_ok=True)

    url = row.Image
#     print(f"Downloading {url} to {filename}")

    try:
        r = requests.get(url, allow_redirects=True)
        with open(filename, 'wb') as f:
            f.write(r.content)
    except:
        print(f'{filename} error')


def cardinality_threshold(column,threshold=0.75,return_categories_list=True):
    #calculate the threshold value using
    #the frequency of instances in column
    threshold_value=int(threshold*len(column))
    #initialize a new list for lower cardinality column
    categories_list=[]
    #initialize a variable to calculate sum of frequencies
    s=0
    #Create a dictionary (unique_category: frequency)
    counts=Counter(column)

    #Iterate through category names and corresponding frequencies after sorting the categories
    #by descending order of frequency
    for i,j in counts.most_common():
        #Add the frequency to the total sum
        s += dict(counts)[i]
        #append the category name to the categories list
        categories_list.append(i)
        #Check if the global sum has reached the threshold value, if so break the loop
        if s >= threshold_value:
            break
      #append the new 'Other' category to list
    categories_list.append('Other')

    #Take all instances not in categories below threshold
```

```
    #that were kept and lump them into the
    #new 'Other' category.
    new_column = column.apply(lambda x: x if x in categories_list else 'Other')

    #Return the transformed column and
    #unique categories if return_categories = True
    if(return_categories_list):
        return new_column,categories_list
    #Return only the transformed column if return_categories=False
    else:
        return new_column
```

Listed Data

This section explores data retrieved from making public eBay API calls. It contains more detailed information compared to the data retrieved from scraping the Teraform website.

See Appendix at end of notebook for exploration of aspects from the Item Specifics section of eBay and for a table explaining all data columns in this project.

In [3]:
```
listed_df = pd.read_csv('listed_data/listed_used_knives.csv')
```

In [4]:
```
listed_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14310 entries, 0 to 14309
Data columns (total 21 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   itemId               14310 non-null  int64  
 1   title                14310 non-null  object 
 2   galleryURL           14309 non-null  object 
 3   viewItemURL          14310 non-null  object 
 4   autoPay              14310 non-null  bool   
 5   postalCode           13942 non-null  object 
 6   sellingStatus        14310 non-null  object 
 7   shippingInfo         14310 non-null  object 
 8   listingInfo          14310 non-null  object 
 9   returnsAccepted      14310 non-null  bool   
 10  condition            14309 non-null  float64
 11  topRatedListing      14310 non-null  bool   
 12  pictureURLLarge      13579 non-null  object 
 13  pictureURLSuperSize  13524 non-null  object 
 14  shipping_cost        14310 non-null  float64
 15  price_in_US          14310 non-null  float64
 16  converted_price      14310 non-null  float64
 17  brand                14310 non-null  object 
 18  cost                 14310 non-null  float64
 19  profit               14310 non-null  float64
 20  ROI                  14310 non-null  float64
dtypes: bool(3), float64(7), int64(1), object(10)
memory usage: 2.0+ MB
```

IQR filtering applied to listed data in below cell

GalleryURL:

pictureURLLarge:

pictureURLSuperSize:

example of a list of pics from the pictureURL column

In [5]:
```
#run this to view example of pictures avaliable in a random row

# df_listed[['galleryURL',
#              'pictureURLLarge',
#              'pictureURLSuperSize',
#              'PictureURL']].sample(1).apply([print])
```

Sold Data: Scraped from the Terapeak website

This is a larger dataset containing listings that go back 2 years instead of 90 days. The data was also filtered to only include sold data of used knives. That means all prices in the dataset reflect the true final sale price of each item.

Terapeak: eBay Research Tools (https://www.ebay.com/help/selling/selling-tools/terapeak-research?id=4853)

In [6]:
```
sold_df = pd.read_csv('terapeak_data/sold_df.csv')
```

In [7]:
```
sold_df.brand.value_counts()
```

Out[7]:
```
case          18918
kershaw       12957
buck          12534
victorinox     9437
spyderco       6046
benchmade      5712
crkt           4276
sog            3006
Name: brand, dtype: int64
```

In [8]:
```
sold_df.drop(['price_in_US',
              'shipping_cost',
              'url', 'cost'],
             axis=1, inplace=True)
```

Combined Data:

Adding used listed data to the dataframe of sold data.

In [9]:
```
#filter for used listed knives
used_listed = listed_df.loc[listed_df['condition'] != 1000]
```

In [10]:
```
cols = ['title','pictureURLLarge','converted_price','brand','profit','ROI']
used_listed2 = used_listed[cols].copy()
used_listed2.dropna(subset=['pictureURLLarge'], inplace=True)
```

In [11]:
```
used_listed2.reset_index(drop=True, inplace=True)
```

In [12]:
```
apply_iqr_filter(used_listed2).describe()
```

Out[12]:

	converted_price	profit	ROI
count	12318.000000	12318.000000	12318.000000
mean	55.731031	25.883789	112.725048
std	46.798688	39.627238	169.038415
min	2.990000	-41.918700	-88.690000
25%	20.412500	-3.860000	-16.820435
50%	38.000000	11.756500	51.304348
75%	77.785000	44.860000	198.826087
max	215.000000	158.786500	690.376087

In [13]:
```
apply_iqr_filter(sold_df).describe()
```

Out[13]:

	converted_price	profit	ROI
count	66341.000000	66341.000000	66341.000000
mean	47.889362	12.903387	41.335537
std	33.605748	25.958237	86.162417
min	0.990000	-45.555000	-96.918426
25%	21.990000	-6.243500	-24.220915
50%	37.950000	5.971300	21.387352
75%	64.700000	25.350000	86.762075
max	163.350000	100.989000	331.140250

In [14]:
```
#combine used listed and used sold data
df1 = pd.concat([sold_df, used_listed2]).copy()
df1['Image'].fillna(df1['pictureURLLarge'], inplace=True)
```

IQR filtering applied to listed data in below cell

In [15]:
```
df = apply_iqr_filter(df1).copy()
```

In [16]:
```
1 df.describe()
```

Out[16]:

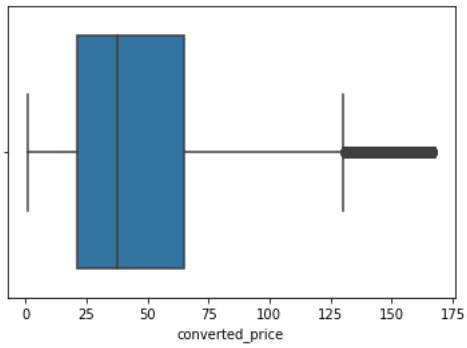
	converted_price	profit	ROI
count	78152.000000	78152.000000	78152.000000
mean	48.309611	14.126540	48.024340
std	34.668644	27.202213	94.605069
min	0.990000	-45.555000	-96.918426
25%	21.500000	-6.026000	-24.183007
50%	37.500000	6.577800	24.352415
75%	65.000000	27.491700	96.069052
max	167.500000	108.810000	371.960784

In [17]:
```
1 delta_listed = ((len(df) - len(df1))/len(df))*100
2 print(delta_listed)
```

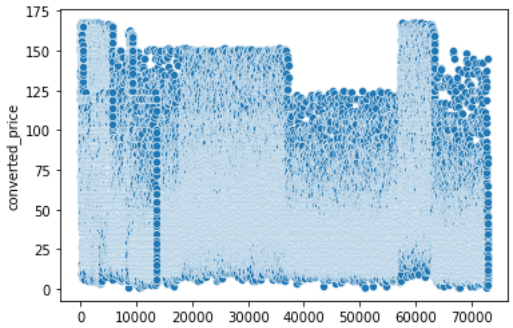
```
-10.636963865288157
```

In [18]:
```
1 sns.boxplot(x=df['converted_price'])
```

Out[18]: <AxesSubplot:xlabel='converted_price'>



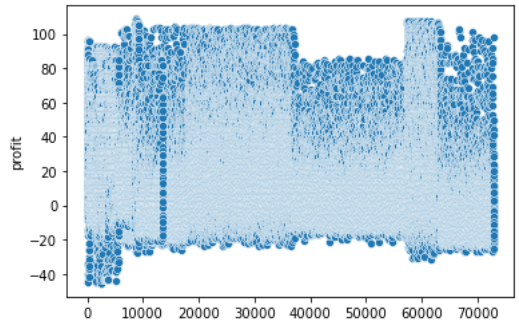
In [19]:
```
1 sns.scatterplot(x=df.index,y='converted_price', data=df);
```



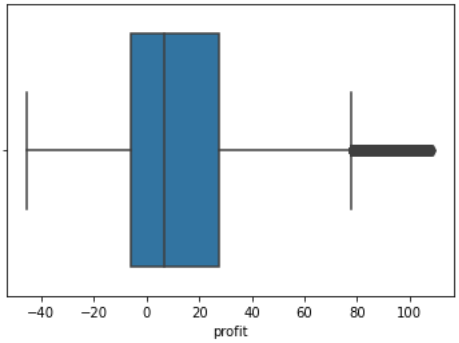
In [20]:
```
1 df['converted_price'].describe()
```

Out[20]:
```
count    78152.000000
mean        48.309611
std         34.668644
min          0.990000
25%         21.500000
50%         37.500000
75%         65.000000
max        167.500000
Name: converted_price, dtype: float64
```

In [21]:
```
1 sns.scatterplot(x=df.index,y='profit', data=df);
```



In [22]:
```
1 sns.boxplot(x=df['profit']);
```



In [23]:
```
1 start_day = '10.30.2022'
2 end_day = '01.27.2023'
3
4 # Convert start / end dates to datetime
5 start_day = pd.to_datetime(start_day)
6 end_day = pd.to_datetime(end_day)
```

In [24]:
```
1 df_sold = apply_iqr_filter(sold_df).copy()
```

In [25]:
```
1 df_sold['date_sold'] = pd.to_datetime(df_sold['date_sold'])
```

In [26]:
```
1 freq_sold_90days = df_sold.loc[df_sold['date_sold'].between(start_day, end_day), 'brand'].value_counts()/90
```

In [27]:
```
1 freq_sold_90days
```

Out[27]:
```
victorinox    32.855556
case          25.922222
buck          18.255556
kershaw       17.300000
spyderco       8.222222
benchmade      7.166667
crkt           6.677778
sog            4.511111
Name: brand, dtype: float64
```

Data Exploration of Sold Data

-a few thousand sold listed knives were also added tot the dataframe for a total fo over 76K knives! The number of listed knives added was small enough compared to the sold data to not skew the continous target functions very much or standard deviations.

Exploring Barplots to Visualize Central Tendencies of Full Dataset

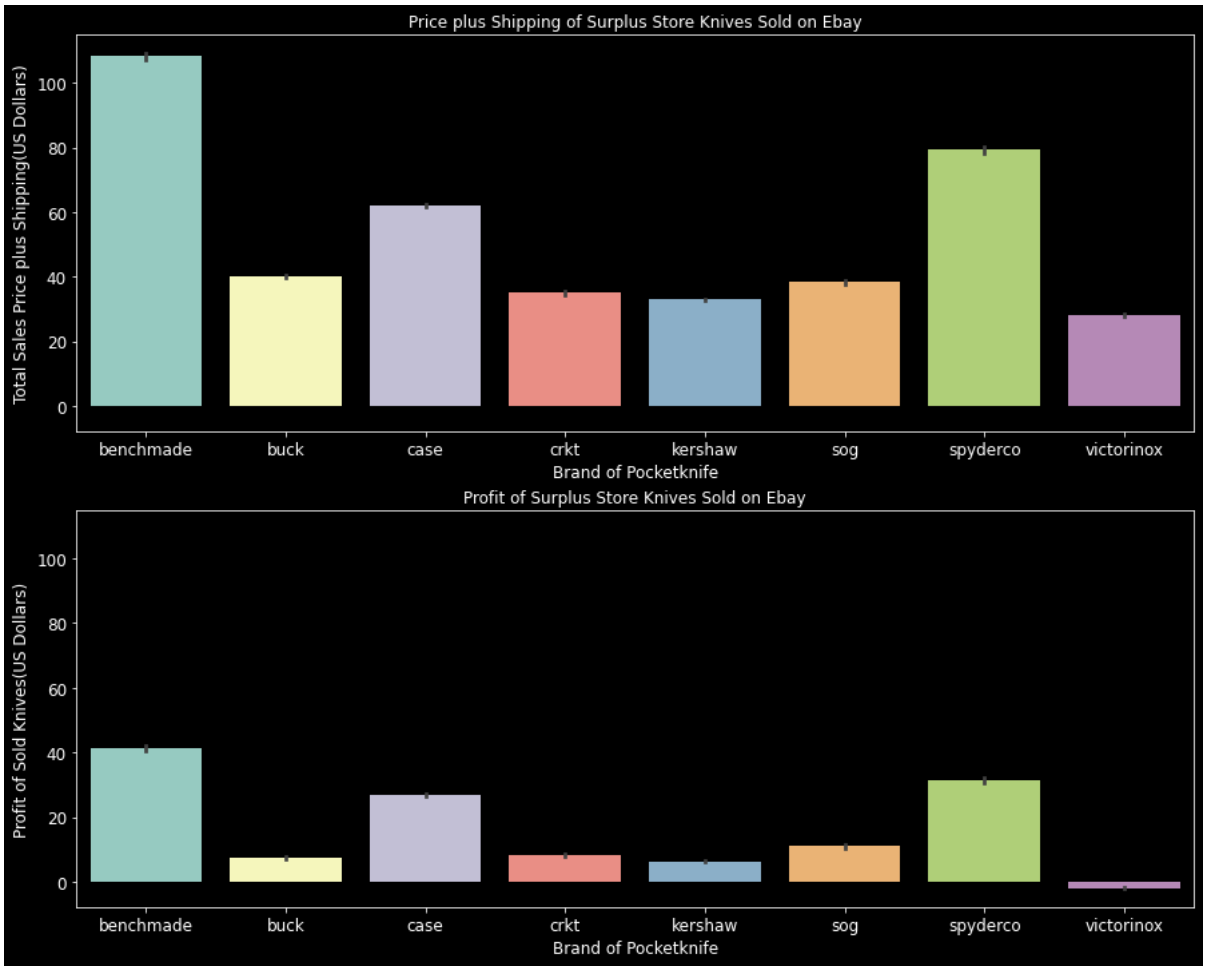
> The full dataset of 76K knives should be a large enough sample set to be a good apprixmation for the real sold value (shipping included) of all used pocketknives sold in the past 2 years for each approriate brand of pocketknive.

> The plots for Price Sold and profit paint a picture clearly showing Benchmade knives as the highest priced and most profitable knives. Spyderco prices/profits follow, with case knives not far behind. However, just sold price and profits don't paint the whole picture. The graph below the price/profits graph show another feature that was created to account for the cost of the knives as well: Return on Investment (% US dollars).

> The barplot by brand for ROI% reveals that one could expect lower risk and greater returns from investing in the cheaper Case brand knives when compared to the more profitable/costly Spyerdco or Benchmade knives. Case knives offer the best return on investment out of all 8 brands of knives tested.

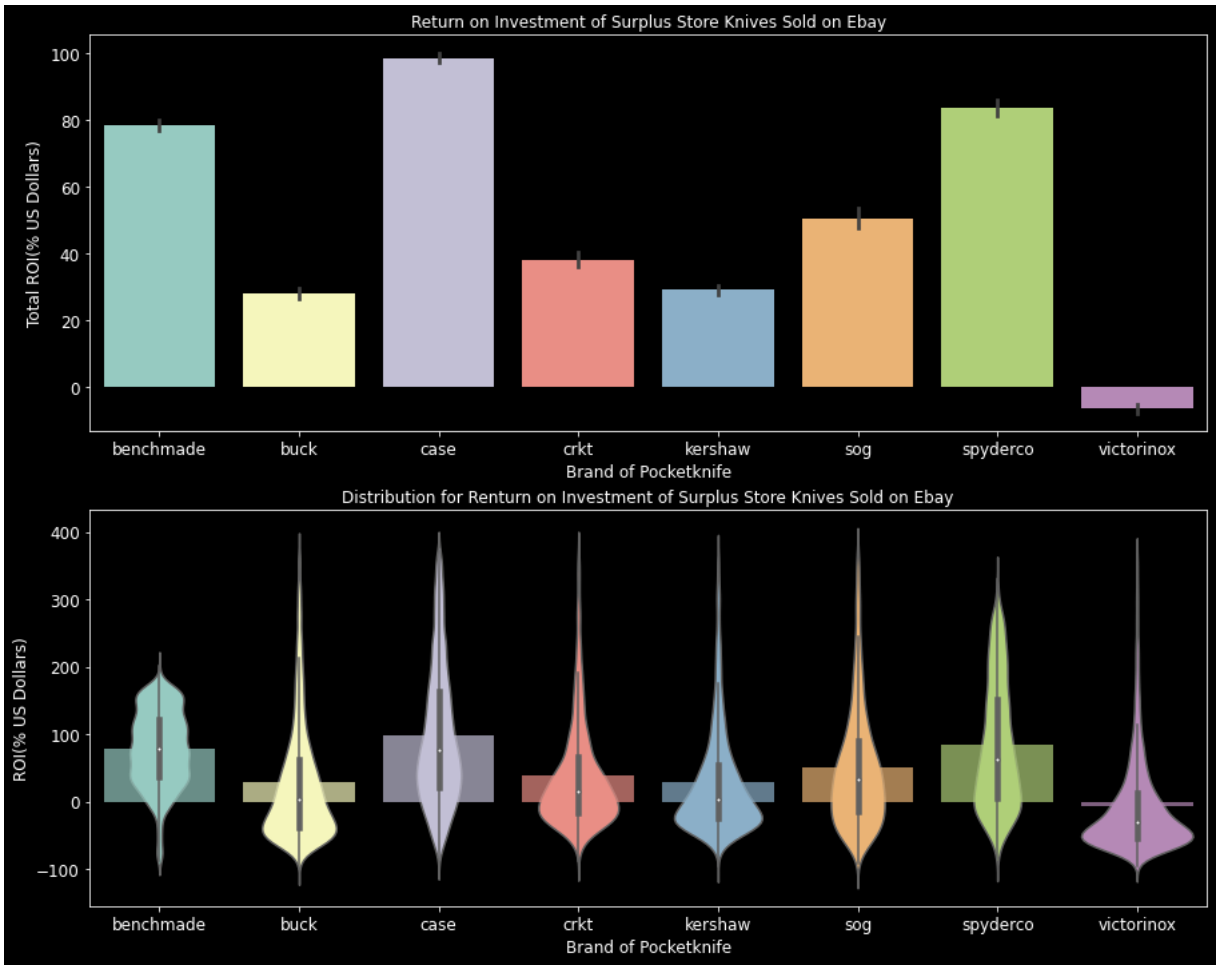
In [28]:
```
from matplotlib import style
style.use('dark_background')
```

In [29]:
```
#converted price == listed price(US dollars) + total shipping cost (US dollars)

fig, axes  = plt.subplots(figsize=(15,12),nrows=2, sharey=True)
sns.barplot(x= 'brand', y='converted_price', ax=axes[0],data=df)
sns.barplot(x= 'brand', y='profit', ax=axes[1],data=df)

axes[0].title.set_text("Price plus Shipping of Surplus Store Knives Sold on Ebay")
axes[0].set_xlabel('Brand of Pocketknife', fontsize=12)
axes[0].set_ylabel('Total Sales Price plus Shipping(US Dollars)', fontsize=12)
axes[0].tick_params(axis='both', labelsize=12)

axes[1].title.set_text("Profit of Surplus Store Knives Sold on Ebay")
axes[1].set_xlabel('Brand of Pocketknife', fontsize=12)
axes[1].set_ylabel('Profit of Sold Knives(US Dollars)', fontsize=12)
axes[1].tick_params(axis='both', labelsize=12)


plt.savefig('images/price_profit_graphs.png');
```

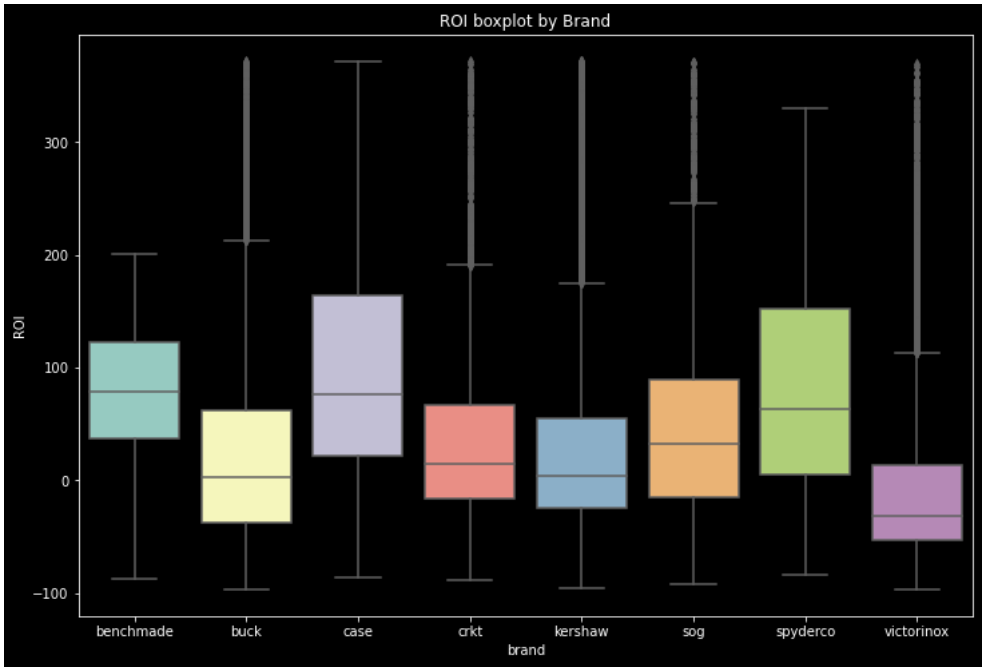


In [30]:

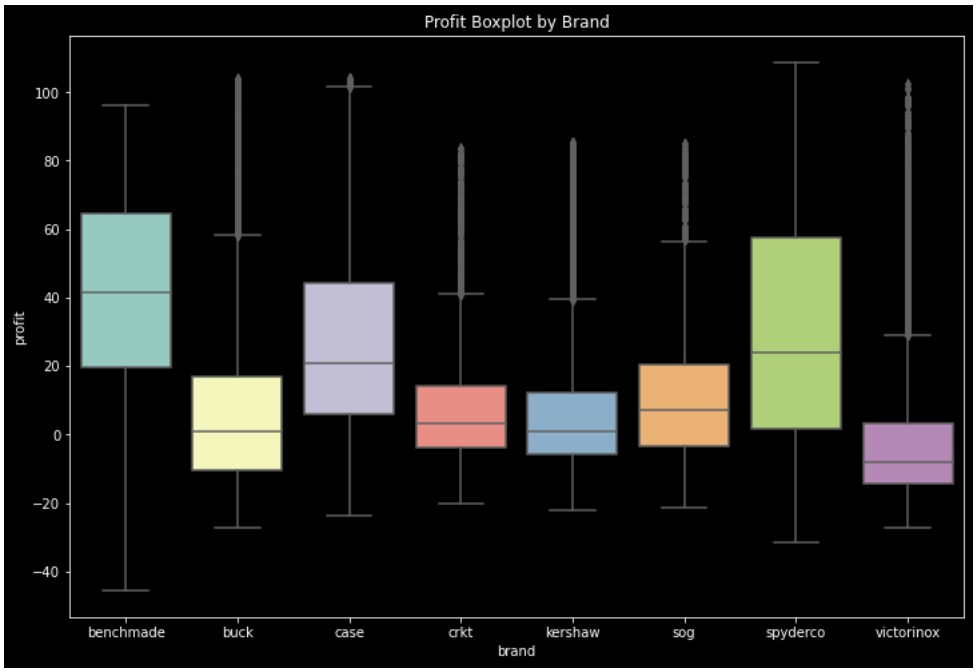
```
#converted price == listed price(US dollars) + total shipping cost (US dollars)

fig, axes  = plt.subplots(figsize=(15,12),nrows=2)
sns.barplot(x= 'brand', y='ROI', ax=axes[0],data=df)
sns.barplot(x= 'brand', y='ROI',alpha=0.7, ax=axes[1],data=df)
sns.violinplot(x= 'brand', y='ROI',data=df)
axes[0].title.set_text("Return on Investment of Surplus Store Knives Sold on Ebay")
axes[0].set_xlabel('Brand of Pocketknife', fontsize=12)
axes[0].set_ylabel('Total ROI(% US Dollars)', fontsize=12)
axes[0].tick_params(axis='both', labelsize=12)

axes[1].title.set_text("Distribution for Renturn on Investment of Surplus Store Knives Sold on Ebay")
axes[1].set_xlabel('Brand of Pocketknife', fontsize=12)
axes[1].set_ylabel('ROI(% US Dollars)', fontsize=12)
axes[1].tick_params(axis='both', labelsize=12)


plt.savefig('images/ROI_graphs.png');
```



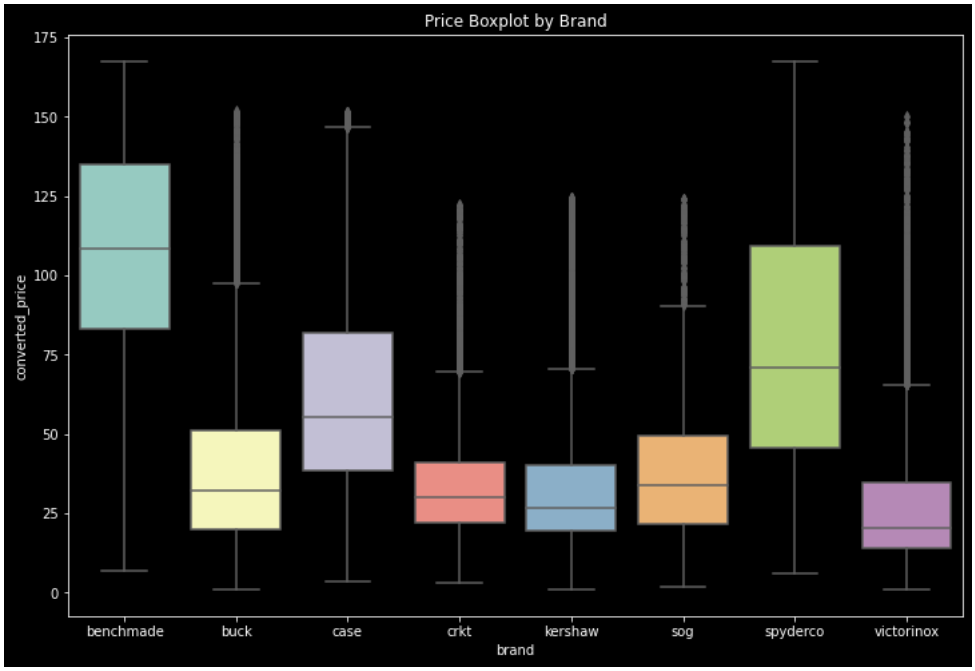
In [31]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x= 'brand', y='ROI',data=df)
plt.title("ROI boxplot by Brand")
plt.show();
```



In [32]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x= 'brand', y='profit',data=df)
plt.title("Profit Boxplot by Brand")
plt.show();
```



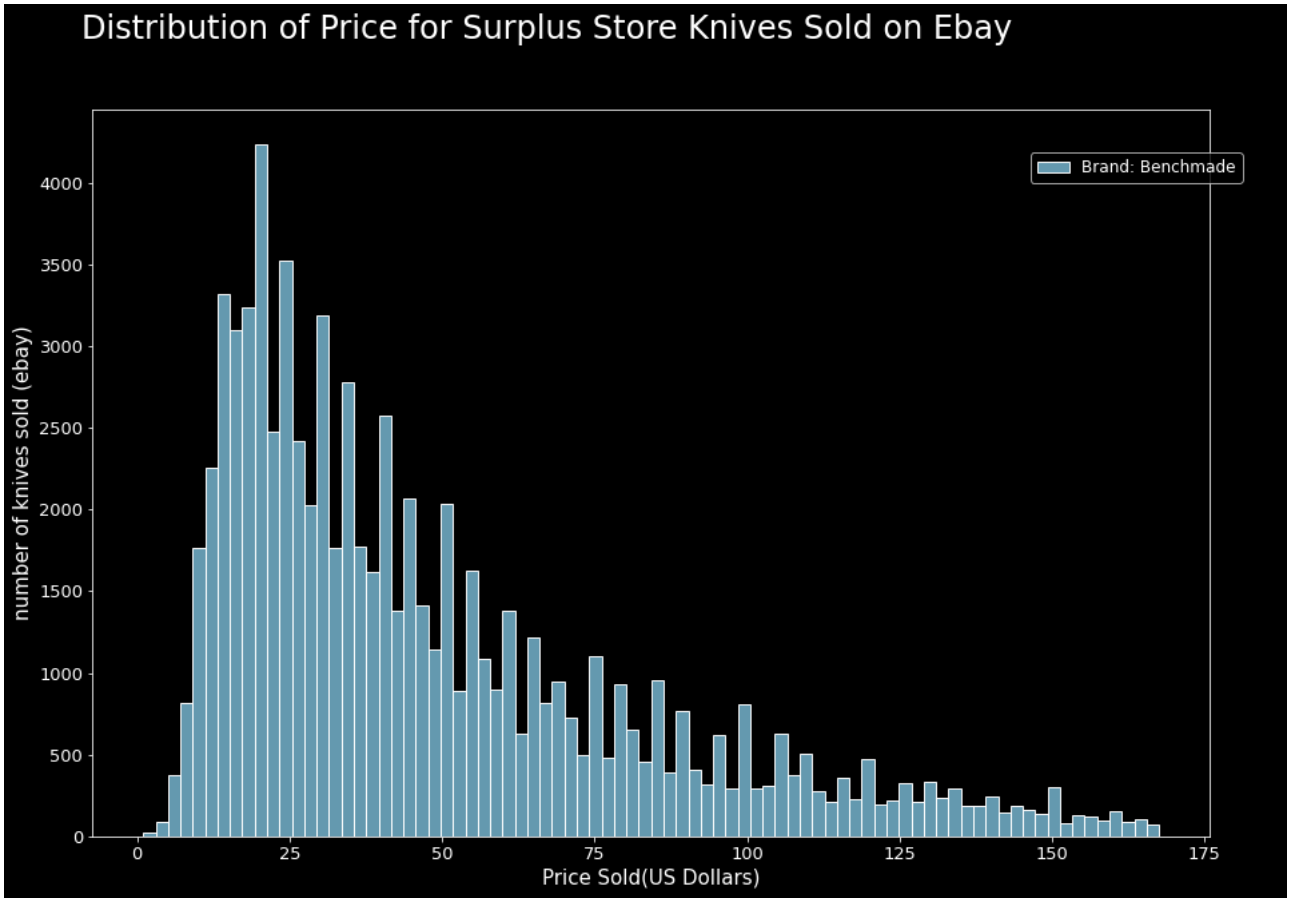
In [33]:
```
plt.figure(figsize=(12,8))
sns.boxplot(x= 'brand', y='converted_price',data=df)
plt.title("Price Boxplot by Brand")
plt.show();
```



In [34]:
```
bench_sold = df.loc[df['brand'] == 'benchmade'].copy()
buck_sold = df.loc[df['brand'] == 'buck'].copy()
case_sold = df.loc[df['brand'] == 'case'].copy()
crkt_sold = df.loc[df['brand'] == 'crkt'].copy()
kershaw_sold = df.loc[df['brand'] == 'kershaw'].copy()
sog_sold = df.loc[df['brand'] == 'sog'].copy()
spyd_sold = df.loc[df['brand'] == 'spyderco'].copy()
vict_sold = df.loc[df['brand'] == 'victorinox'].copy()
```

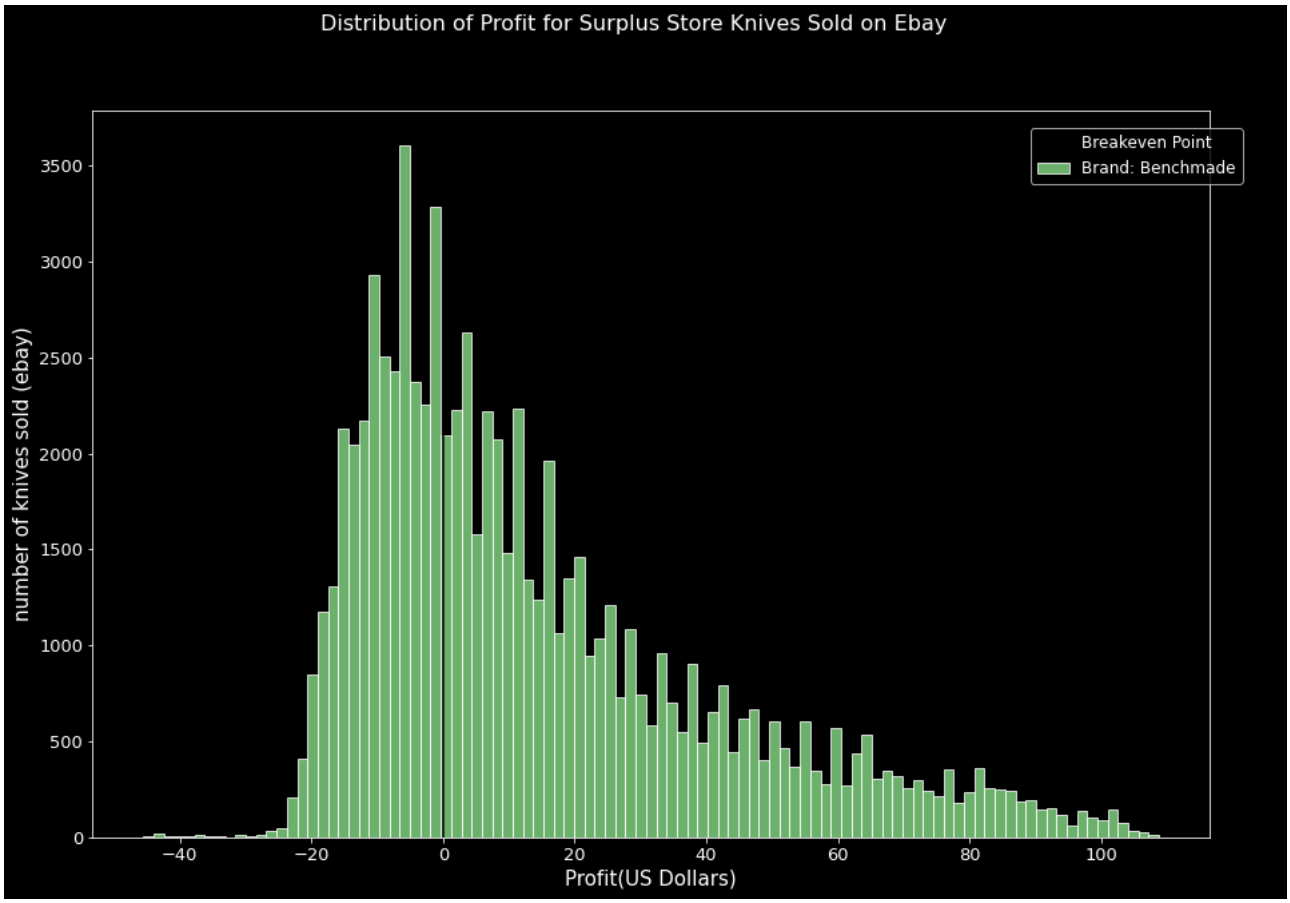
Distribution of Prices, Profits, and ROI

In [35]:
```
fig, axes  = plt.subplots(figsize=(15,10), ncols=1)

sns.histplot(df['converted_price'], color='skyblue', label='Brand: Benchmade');
axes.set_xlabel('Price Sold(US Dollars)', fontsize=15)
axes.set_ylabel('number of knives sold (ebay)', fontsize=15)
axes.tick_params(axis='both', labelsize=13)


fig.suptitle("Distribution of Price for Surplus Store Knives Sold on Ebay", fontsize=24, x=0.44)
fig.legend(loc=(.8, .80), fontsize='large')
plt.savefig('images/dist_price_graph.png');
```



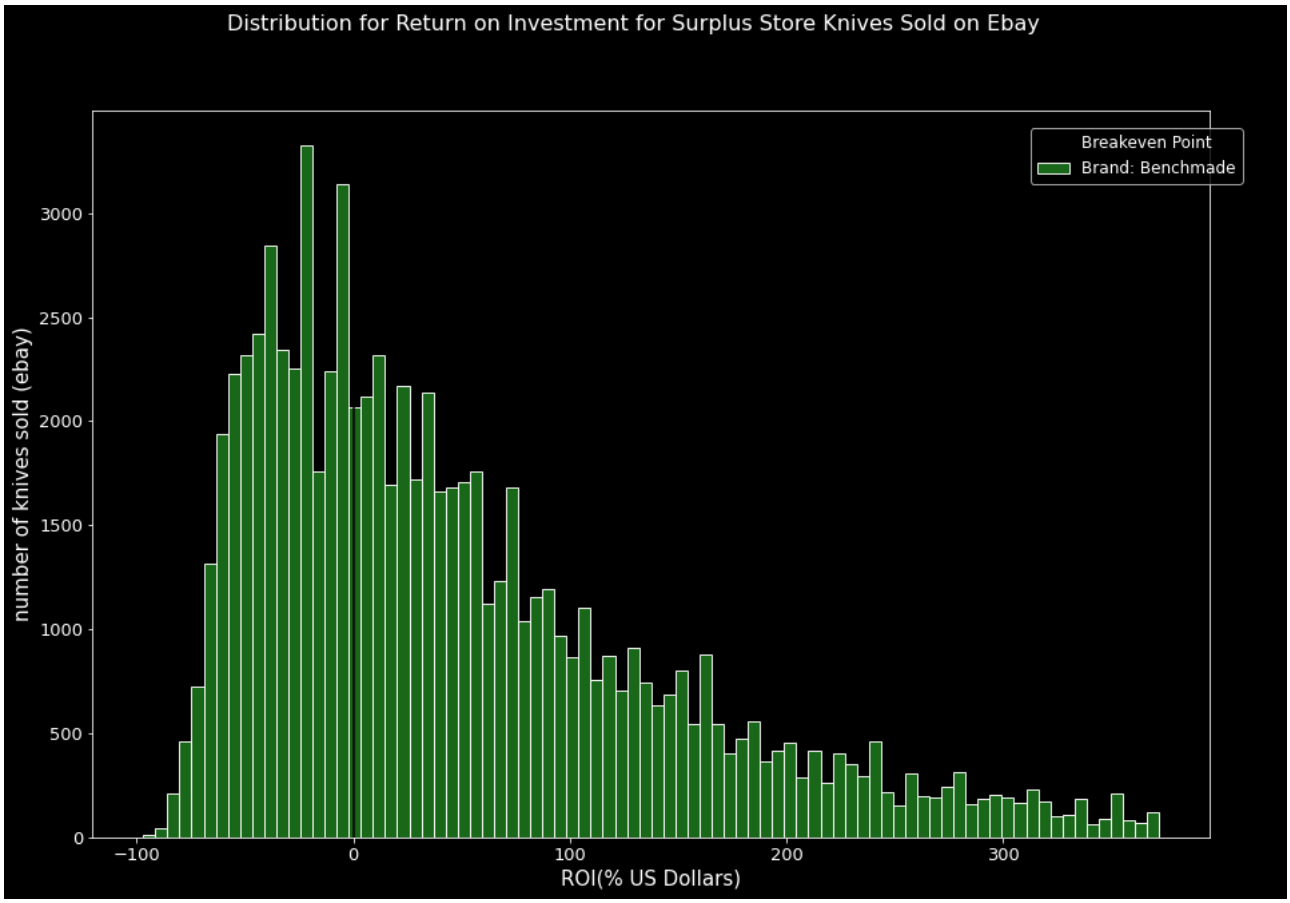
In [36]:

```
fig, axes  = plt.subplots(figsize=(15,10), ncols=1)

sns.histplot(df['profit'], color='lightgreen', label='Brand: Benchmade');
axes.set_xlabel('Profit(US Dollars)', fontsize=15)
axes.set_ylabel('number of knives sold (ebay)', fontsize=15)
axes.tick_params(axis='both', labelsize=13)

axes.axvline(x = 0, color = 'black', label= 'Breakeven Point')
fig.suptitle("Distribution of Profit for Surplus Store Knives Sold on Ebay", fontsize=16)
fig.legend(loc=(.8, .80), fontsize='large')
plt.savefig('images/dist_profit_graph.png');
```



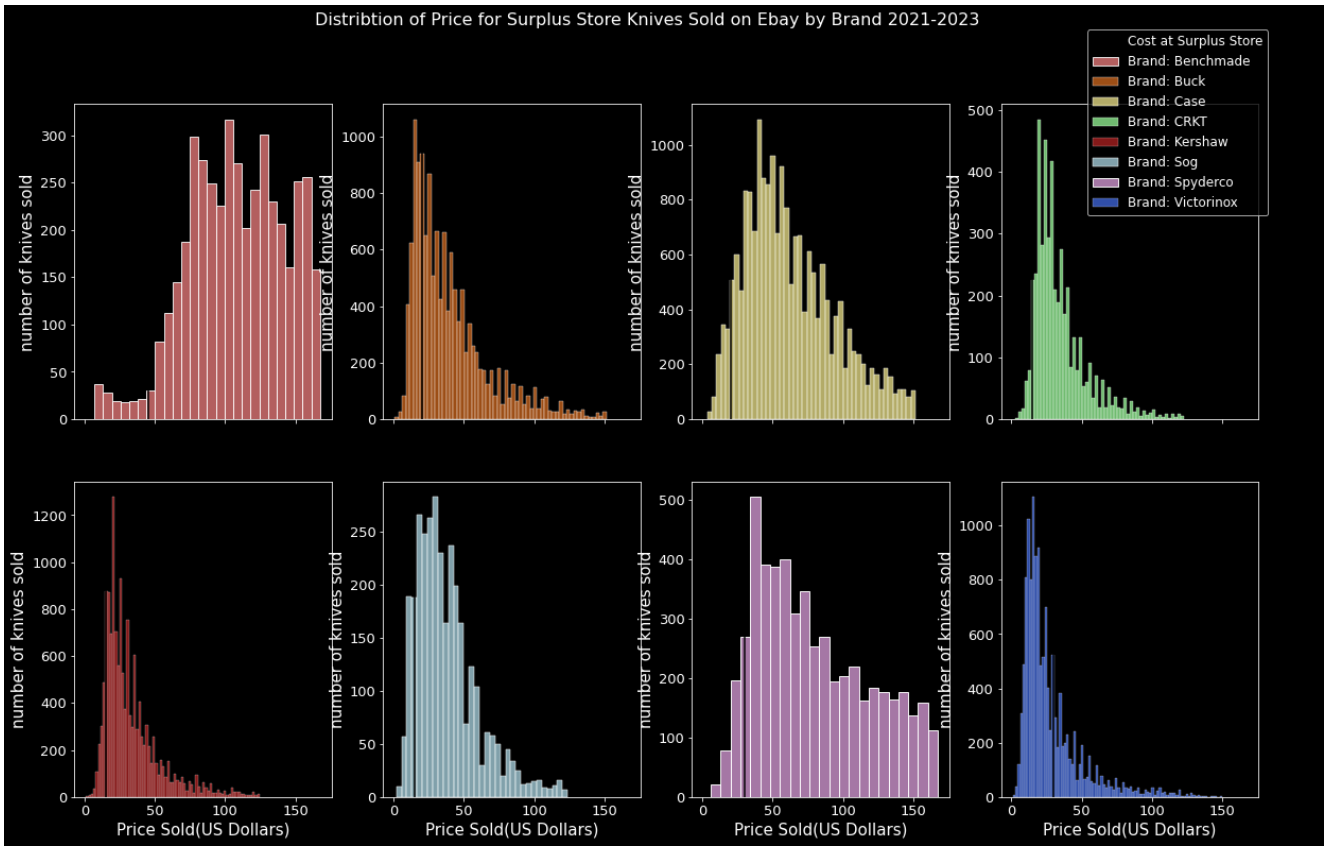
In [37]:

```
fig, axes  = plt.subplots(figsize=(15,10), ncols=1)

sns.histplot(df['ROI'], color='forestgreen', label='Brand: Benchmade');
axes.set_xlabel('ROI(% US Dollars)', fontsize=15)
axes.set_ylabel('number of knives sold (ebay)', fontsize=15)
axes.tick_params(axis='both', labelsize=13)

axes.axvline(x = 0, color = 'black', label= 'Breakeven Point')
fig.suptitle("Distribution for Return on Investment for Surplus Store Knives Sold on Ebay", fontsize=16)
fig.legend(loc=(.8, .80), fontsize='large')
plt.savefig('images/dist_ROI_graph.png');
```



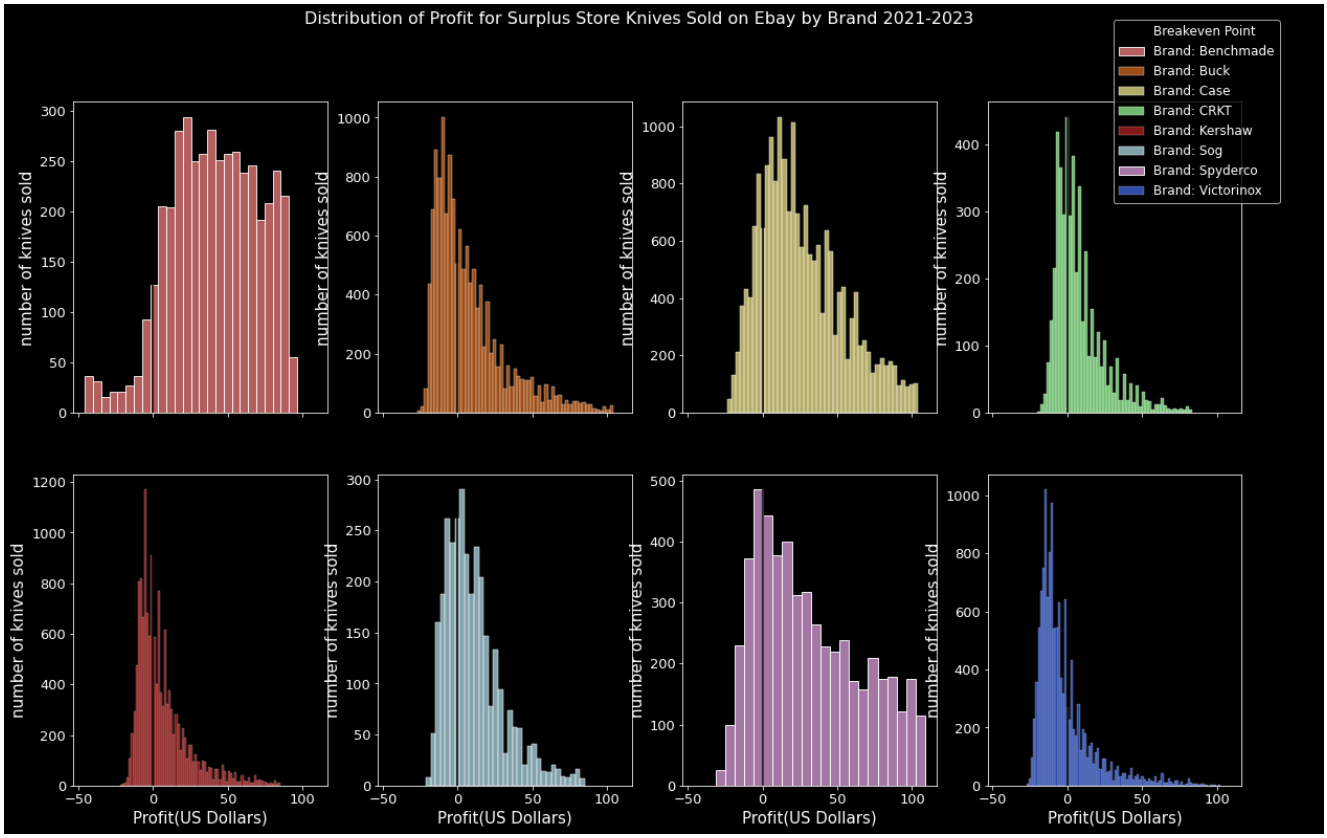
In [38]:

```
fig, axes  = plt.subplots(figsize=(20,12), ncols=4, nrows=2, sharex=True)
sns.histplot(bench_sold['converted_price'], ax=axes[0][0], color='lightcoral', label='Brand: Benchmade')
sns.histplot(buck_sold['converted_price'], ax=axes[0][1], color='chocolate', label='Brand: Buck')
sns.histplot(case_sold['converted_price'], ax=axes[0][2],  color='khaki', label='Brand: Case')
sns.histplot(crkt_sold['converted_price'], ax=axes[0][3], color='palegreen', label='Brand: CRKT')
sns.histplot(kershaw_sold['converted_price'], ax=axes[1][0], color='firebrick', label='Brand: Kershaw')
sns.histplot(sog_sold['converted_price'], ax=axes[1][1],  color='lightblue', label='Brand: Sog')
sns.histplot(spyd_sold['converted_price'], ax=axes[1][2],  color='plum', label='Brand: Spyderco')
sns.histplot(vict_sold['converted_price'], ax=axes[1][3],  color='royalblue', label='Brand: Victorinox')


for n in range(8):
    row = n//4
    col = n%4
    axes[row][col].set_xlabel('Price Sold(US Dollars)', fontsize=15)
    axes[row][col].set_ylabel('number of knives sold', fontsize=15)
    axes[row][col].tick_params(axis='both', labelsize=13)

axes[0][0].axvline(x = 45, color = 'black', label= 'Cost at Surplus Store')
axes[0][1].axvline(x = 20, color = 'black')
axes[0][2].axvline(x = 20, color = 'black')
axes[0][3].axvline(x = 15, color = 'black')
axes[1][0].axvline(x = 15, color = 'black')
axes[1][1].axvline(x = 15, color = 'black')
axes[1][2].axvline(x = 30, color = 'black')
axes[1][3].axvline(x = 30, color = 'black')
fig.suptitle("Distribtion of Price for Surplus Store Knives Sold on Ebay by Brand 2021-2023", fontsize=16)
fig.legend(loc=(.82, .75), fontsize='large')
plt.savefig('images/4x4_price_graphs.png');
```



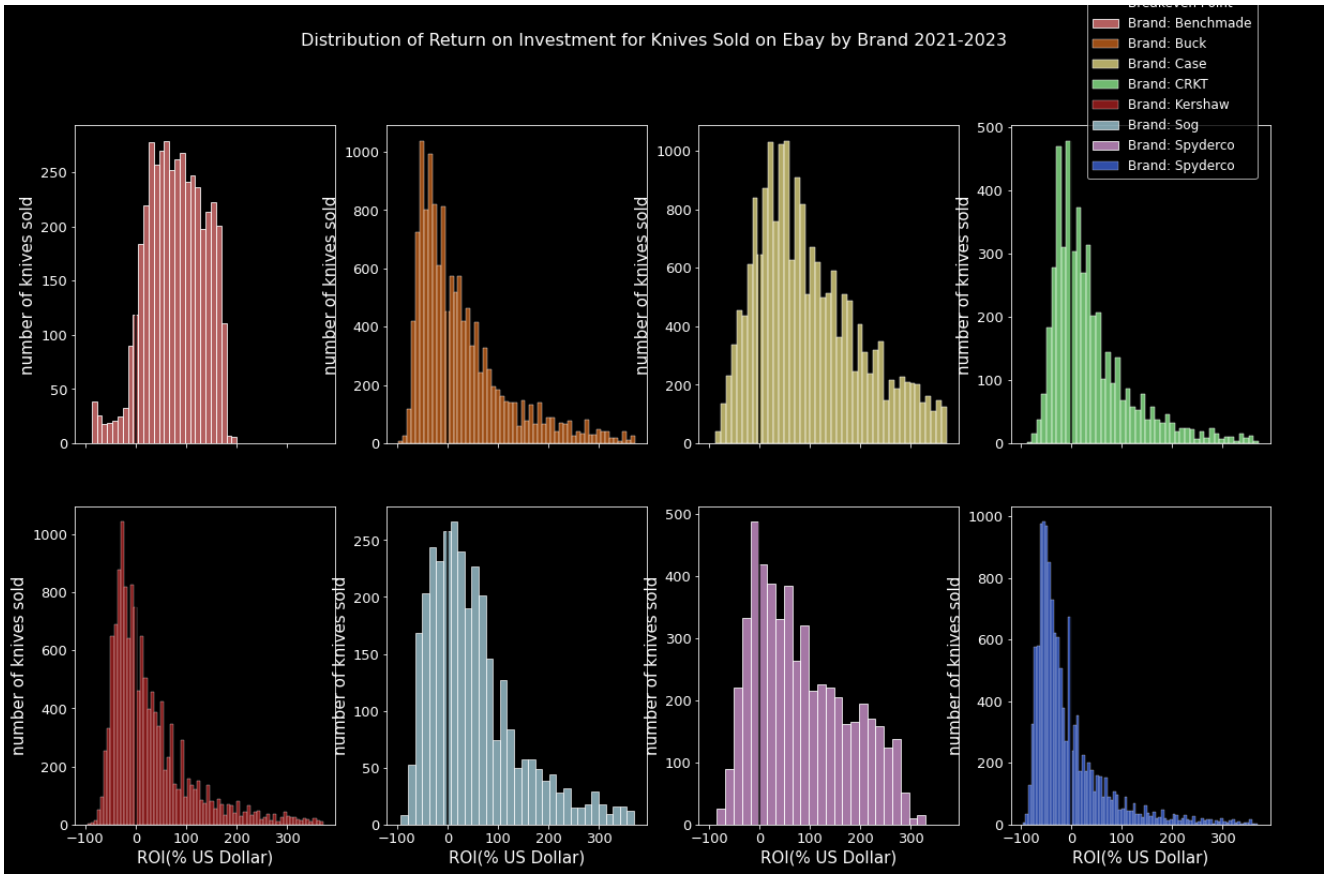
In [39]:

```
fig, axes  = plt.subplots(figsize=(20,12), ncols=4, nrows=2, sharex=True)
sns.histplot(bench_sold['profit'], ax=axes[0][0], color='lightcoral', label='Brand: Benchmade')
sns.histplot(buck_sold['profit'], ax=axes[0][1], color='chocolate', label='Brand: Buck')
sns.histplot(case_sold['profit'], ax=axes[0][2],  color='khaki', label='Brand: Case')
sns.histplot(crkt_sold['profit'], ax=axes[0][3], color='palegreen', label='Brand: CRKT')
sns.histplot(kershaw_sold['profit'], ax=axes[1][0], color='firebrick', label='Brand: Kershaw')
sns.histplot(sog_sold['profit'], ax=axes[1][1],  color='lightblue', label='Brand: Sog')
sns.histplot(spyd_sold['profit'], ax=axes[1][2],  color='plum', label='Brand: Spyderco')
sns.histplot(vict_sold['profit'], ax=axes[1][3],  color='royalblue', label='Brand: Victorinox')

for n in range(8):
    row = n//4
    col = n%4
    axes[row][col].set_xlabel('Profit(US Dollars)', fontsize=15)
    axes[row][col].set_ylabel('number of knives sold', fontsize=15)
    axes[row][col].tick_params(axis='both', labelsize=13)

axes[0][0].axvline(x = 0, color = 'black', label= 'Breakeven Point')
axes[0][1].axvline(x = 0, color = 'black')
axes[0][2].axvline(x = 0, color = 'black')
axes[0][3].axvline(x = 0, color = 'black')
axes[1][0].axvline(x = 0, color = 'black')
axes[1][1].axvline(x = 0, color = 'black')
axes[1][2].axvline(x = 0, color = 'black')
axes[1][3].axvline(x = 0, color = 'black')
fig.suptitle("Distribution of Profit for Surplus Store Knives Sold on Ebay by Brand 2021-2023", fontsize=16)
fig.legend(loc=(.84, .76), fontsize='large')
plt.savefig('images/4x4_profit_graphs.png');
```



In [40]:

```
fig, axes  = plt.subplots(figsize=(20,12), ncols=4, nrows=2, sharex=True)
sns.histplot(bench_sold['ROI'], ax=axes[0][0], color='lightcoral', label='Brand: Benchmade')
sns.histplot(buck_sold['ROI'], ax=axes[0][1], color='chocolate', label='Brand: Buck')
sns.histplot(case_sold['ROI'], ax=axes[0][2],  color='khaki', label='Brand: Case')
sns.histplot(crkt_sold['ROI'], ax=axes[0][3], color='palegreen', label='Brand: CRKT')
sns.histplot(kershaw_sold['ROI'], ax=axes[1][0], color='firebrick', label='Brand: Kershaw')
sns.histplot(sog_sold['ROI'], ax=axes[1][1],  color='lightblue', label='Brand: Sog')
sns.histplot(spyd_sold['ROI'], ax=axes[1][2],  color='plum', label='Brand: Spyderco')
sns.histplot(vict_sold['ROI'], ax=axes[1][3],  color='royalblue', label='Brand: Spyderco')


for n in range(8):
    row = n//4
    col = n%4
    axes[row][col].set_xlabel('ROI(% US Dollar)', fontsize=15)
    axes[row][col].set_ylabel('number of knives sold', fontsize=15)
    axes[row][col].tick_params(axis='both', labelsize=13)

axes[0][0].axvline(x = 0, color = 'black', label= 'Breakeven Point')
axes[0][1].axvline(x = 0, color = 'black')
axes[0][2].axvline(x = 0, color = 'black')
axes[0][3].axvline(x = 0, color = 'black')
axes[1][0].axvline(x = 0, color = 'black')
axes[1][1].axvline(x = 0, color = 'black')
axes[1][2].axvline(x = 0, color = 'black')
axes[1][3].axvline(x = 0, color = 'black')
fig.suptitle("Distribution of Return on Investment for Knives Sold on Ebay by Brand 2021-2023", fontsize=16)
fig.legend(loc=(.82, .8), fontsize='large')
plt.savefig('images/4x4_ROI_graphs.png');
```



In [41]:
```
display(bench_sold['brand'].values[0])
display(bench_sold[['converted_price',
                    'profit', 'ROI']].describe())

display(buck_sold['brand'].values[0])
display(buck_sold[['converted_price',
                    'profit', 'ROI']].describe())

display(case_sold['brand'].values[0])
display(case_sold[['converted_price',
                    'profit', 'ROI']].describe())


display(crkt_sold['brand'].values[0])
display(crkt_sold[['converted_price',
                    'profit', 'ROI']].describe())


display(kershaw_sold['brand'].values[0])
display(kershaw_sold[['converted_price',
                    'profit', 'ROI']].describe())


display(sog_sold['brand'].values[0])
display(sog_sold[['converted_price',
                    'profit', 'ROI']].describe())


display(spyd_sold['brand'].values[0])
display(spyd_sold[['converted_price',
                    'profit', 'ROI']].describe())


display(vict_sold['brand'].values[0])
display(vict_sold[['converted_price',
                    'profit', 'ROI']].describe())
```

'benchmade'

	converted_price	profit	ROI
count	4341.000000	4341.000000	4341.000000
mean	108.255895	41.358061	78.316775
std	34.194618	29.743334	56.535092
min	6.990000	-45.555000	-87.330625
25%	83.000000	19.521000	36.866856
50%	108.490000	41.445000	78.682720
75%	135.140000	64.717500	122.634561
max	167.500000	96.315600	200.657500

'buck'

	converted_price	profit	ROI
count	13512.000000	13512.000000	13512.000000
mean	40.055698	7.447602	28.078801
std	27.275225	23.893729	88.557249
min	0.990000	-27.088700	-96.918426
25%	19.990000	-10.297700	-37.745975
50%	32.230000	0.751300	2.688014
75%	51.000000	17.116000	62.506351
max	151.580000	103.924600	371.823256

'case'

	converted_price	profit	ROI
count	19480.000000	19480.000000	19480.000000
mean	62.081364	26.663020	98.462781
std	31.743499	27.701883	102.816450
min	3.500000	-23.608700	-86.760870
25%	38.420000	5.936500	21.333095
50%	55.525000	20.770000	76.799130
75%	81.750000	44.251300	164.517352
max	151.500000	103.855000	371.574240

```
'crkt'
```

	converted_price	profit	ROI
count	4810.000000	4810.000000	4810.000000
mean	34.980089	8.124840	38.036433
std	19.148961	16.914651	78.027248
min	3.050000	-20.296500	-88.437908
25%	21.892500	-3.766500	-16.601307
50%	29.950000	3.232650	14.673203
75%	41.000000	14.137800	66.653333
max	122.500000	83.625000	371.588333

```
'kershaw'
```

	converted_price	profit	ROI
count	14190.000000	14190.000000	14190.000000
mean	32.939050	6.252555	28.995759
std	20.114913	17.649219	79.900759
min	1.140000	-21.958200	-95.678431
25%	19.500000	-5.576100	-24.590523
50%	26.490000	0.922800	4.058824
75%	40.000000	12.430425	55.424837
max	124.500000	85.365000	371.960784

```
'sog'
```

	converted_price	profit	ROI
count	3224.000000	3224.000000	3224.000000
mean	38.291399	10.989944	50.534513
std	22.340828	19.575009	89.594689
min	1.950000	-21.253500	-92.607843
25%	21.695000	-3.375000	-14.705882
50%	33.990000	7.260750	32.357516
75%	49.275000	20.541300	89.542484
max	124.000000	84.930000	370.065359

```
'spyderco'
```

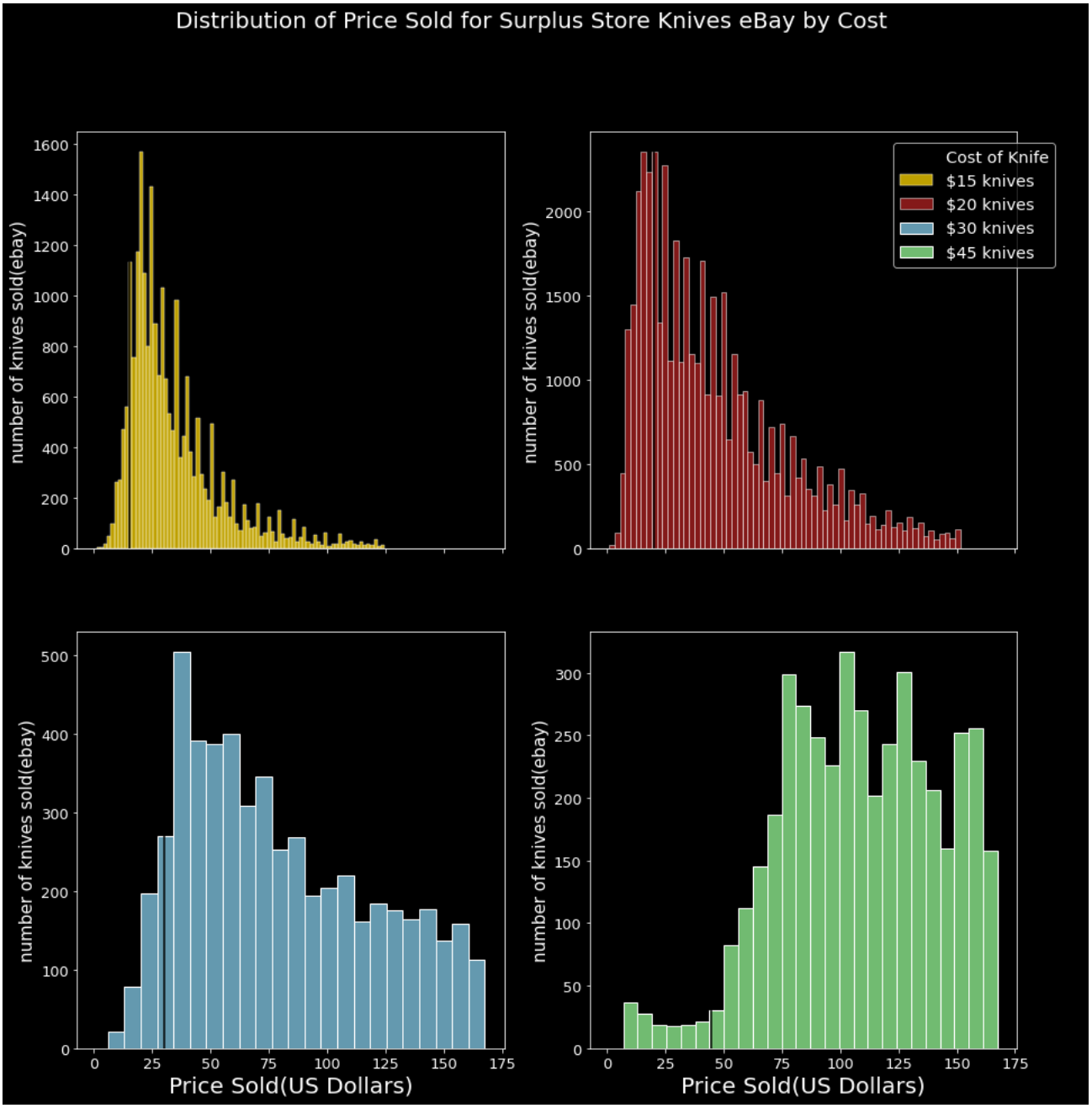
	converted_price	profit	ROI
count	5315.000000	5315.000000	5315.000000
mean	79.270337	31.363509	83.738902
std	40.021260	34.875399	93.413481
min	5.990000	-31.425000	-84.208182
25%	45.445000	1.830750	5.428182
50%	70.750000	23.820000	63.339921
75%	109.350000	57.636900	152.173913
max	167.500000	108.810000	329.727273

```
'victorinox'
```

	converted_price	profit	ROI
count	13280.000000	13280.000000	13280.000000
mean	28.203590	-1.918559	-6.523438
std	21.856962	19.145329	72.998330
min	1.040000	-27.045200	-96.762791
25%	14.137500	-14.343500	-53.340608
50%	20.500000	-7.940000	-31.520572
75%	34.730000	3.100000	13.402609
max	149.990000	102.541300	369.043478

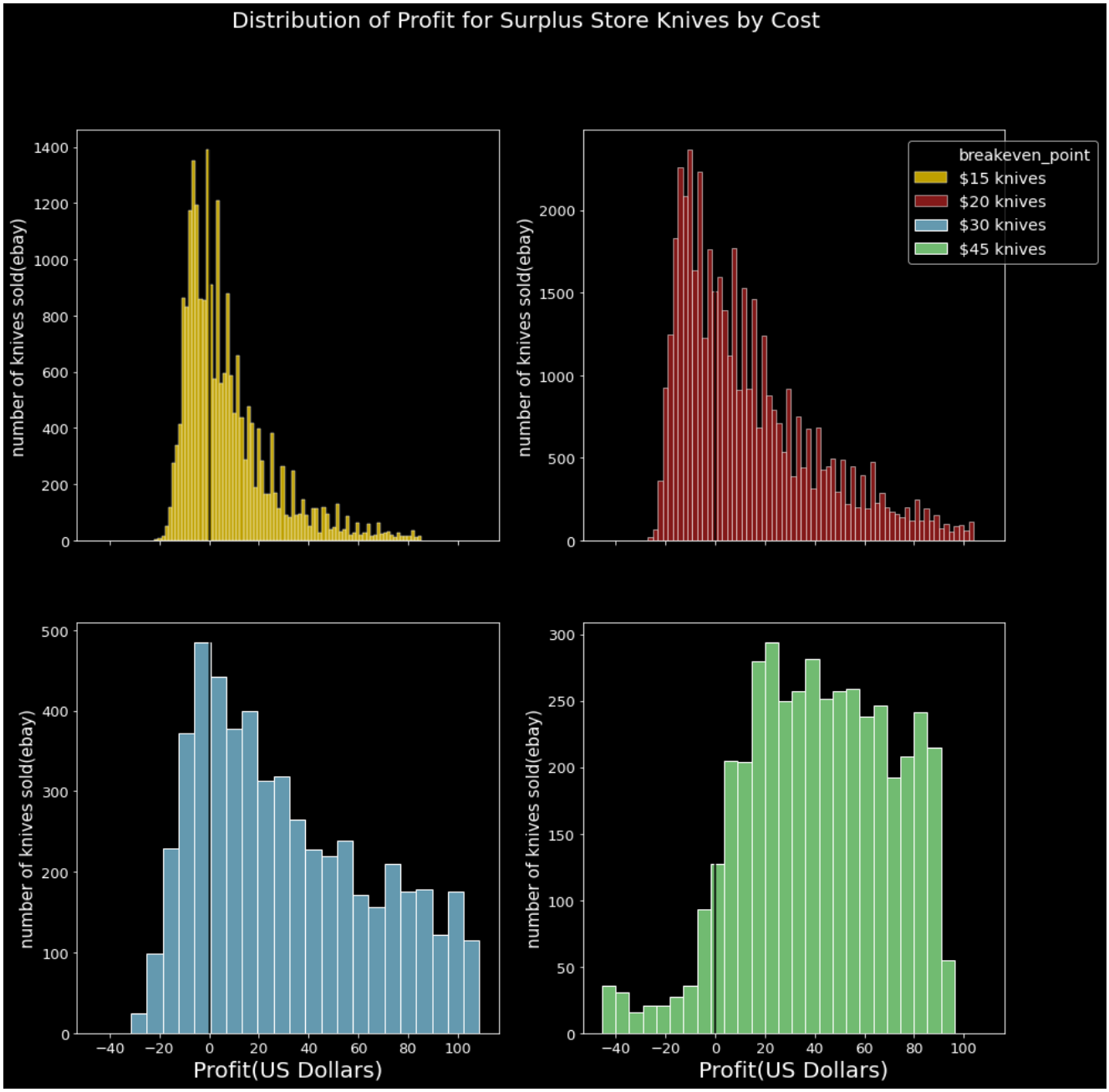
In [42]:

```
df_45 = df.loc[df['brand'] == 'benchmade']
df_30 = df.loc[df['brand'] == 'spyderco']
df_20 = df.loc[(df['brand'] == 'buck') | (df['brand'] == 'case') | (df['brand'] == 'victorinox')]
df_15 = df.loc[(df['brand'] == 'crkt') | (df['brand'] == 'kershaw') | (df['brand'] == 'sog')]


fig, axes  = plt.subplots(figsize=(15,15), ncols=2, nrows=2, sharex=True)
sns.histplot(df_15['converted_price'], ax=axes[0][0], color='gold', label='$15 knives')
sns.histplot(df_20['converted_price'], ax=axes[0][1], color='firebrick', label='$20 knives')
sns.histplot(df_30['converted_price'], ax=axes[1][0],  color='skyblue', label='$30 knives')
sns.histplot(df_45['converted_price'], ax=axes[1][1], color='palegreen', label='$45 knives')

for n in range(4):
    row = n//2 # n divided by 2 without the remainder
    col = n%2  # just the remainder of n divided by 2
    axes[row][col].set_xlabel('Price Sold(US Dollars)', fontsize=20)
    axes[row][col].set_ylabel('number of knives sold(ebay)', fontsize=15)
    axes[row][col].tick_params(axis='both', labelsize=13)


axes[0][0].axvline(x = 15, color = 'black', label= 'Cost of Knife')
axes[0][1].axvline(x = 20, color = 'black')
axes[1][0].axvline(x = 30, color = 'black')
axes[1][1].axvline(x = 45, color = 'black')
fig.suptitle("Distribution of Price Sold for Surplus Store Knives eBay by Cost", fontsize=20)
fig.legend(loc=(.82, .76), fontsize='x-large')
plt.show();
```


Distribution of Price Sold for Surplus Store Knives eBay by Cost
number of knives sold(ebay)
Cost of Knife
$15 knives
$20 knives
$30 knives
$45 knives
Price Sold(US Dollars)

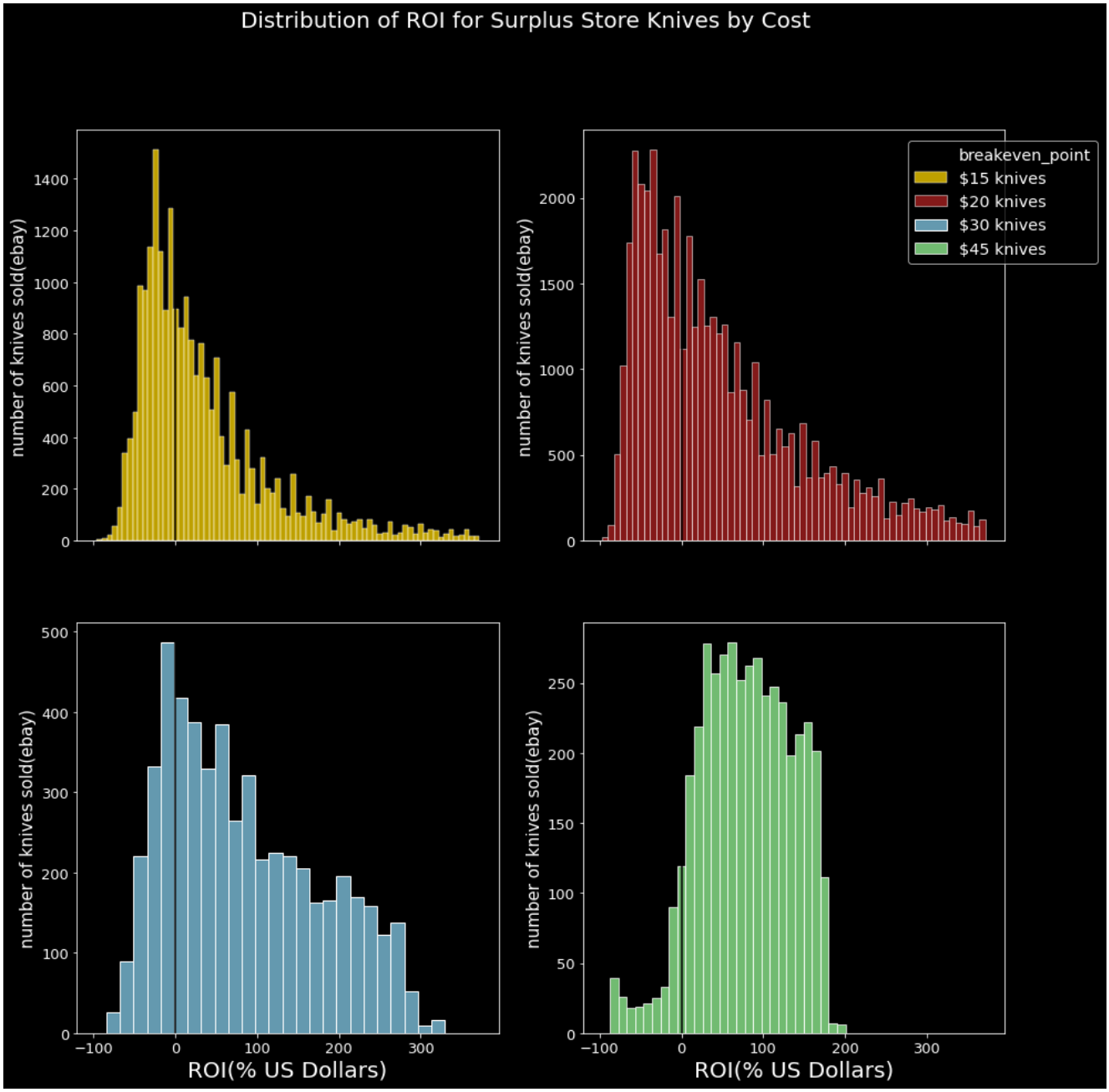
In [43]:

```
fig, axes  = plt.subplots(figsize=(15,15), ncols=2, nrows=2, sharex=True)
sns.histplot(df_15['profit'], ax=axes[0][0], color='gold', label='$15 knives')
sns.histplot(df_20['profit'], ax=axes[0][1], color='firebrick', label='$20 knives')
sns.histplot(df_30['profit'], ax=axes[1][0],  color='skyblue', label='$30 knives')
sns.histplot(df_45['profit'], ax=axes[1][1], color='palegreen', label='$45 knives')

for n in range(4):
    row = n//2 # n divided by 2 without the remainder
    col = n%2  # just the remainder of n divided by 2
    axes[row][col].set_xlabel('Profit(US Dollars)', fontsize=20)
    axes[row][col].set_ylabel('number of knives sold(ebay)', fontsize=15)
    axes[row][col].tick_params(axis='both', labelsize=13)


axes[0][0].axvline(x = 0, color = 'black', label= 'breakeven_point')
axes[0][1].axvline(x = 0, color = 'black')
axes[1][0].axvline(x = 0, color = 'black')
axes[1][1].axvline(x = 0, color = 'black')
fig.suptitle("Distribution of Profit for Surplus Store Knives by Cost", fontsize=20)
fig.legend(loc=(.82, .76), fontsize='x-large')
plt.show();
```



In [44]:

```
fig, axes  = plt.subplots(figsize=(15,15), ncols=2, nrows=2, sharex=True)
sns.histplot(df_15['ROI'], ax=axes[0][0], color='gold', label='$15 knives')
sns.histplot(df_20['ROI'], ax=axes[0][1], color='firebrick', label='$20 knives')
sns.histplot(df_30['ROI'], ax=axes[1][0],  color='skyblue', label='$30 knives')
sns.histplot(df_45['ROI'], ax=axes[1][1], color='palegreen', label='$45 knives')

for n in range(4):
    row = n//2 # n divided by 2 without the remainder
    col = n%2  # just the remainder of n divided by 2
    axes[row][col].set_xlabel('ROI(% US Dollars)', fontsize=20)
    axes[row][col].set_ylabel('number of knives sold(ebay)', fontsize=15)
    axes[row][col].tick_params(axis='both', labelsize=13)


axes[0][0].axvline(x = 0, color = 'black', label= 'breakeven_point')
axes[0][1].axvline(x = 0, color = 'black')
axes[1][0].axvline(x = 0, color = 'black')
axes[1][1].axvline(x = 0, color = 'black')
fig.suptitle("Distribution of ROI for Surplus Store Knives by Cost", fontsize=20)
fig.legend(loc=(.82, .76), fontsize='x-large')
plt.show();
```



Exploration Notes

The Item Specific data was very limited. An attempt was made to incorporate Item Specific details into the sold data scraped from Terapeak, but even with the large increase to pagination required to obtain itemIds, the Item Specific data seems to be taken off the website after 90 days. Therefore, all of the Item Specific data was limited to listed data and was largely used just for visualization purposes to get an idea for what might affect knife prices listed on ebay. The final dataset used for modeling will only include images, titles, prices, and brands. Listed data filtered for used fixed price knives (mages, prices, and titles) were merged into the used data for modeling. A large sample of the listed prices for each model should be a good enough approximation for resale value when compared to the sold data. Nearly 70K samples of knife data came from scraping the Terakpeak website for used, sold knife data. Around 9k samples came from making API calls to eBay for listed data.

Aspect Dataframe (combined Shopping API and Finding API)

Boolean Features

Autopay

Having Autopay turned on or off seems to have some affect on the price of listed knives. Overall, the knives that have autopay turned on seem to appear as being listed slightly higher than knives that are not.

itemId: Unique identifier given to each listing on Ebay.

title: The title of the listing posted by a seller. Should correspond with the brand and model of each knife.

galleryURL: URL for the Gallery thumbnail image (usually around 100X100 or less)

viewItemURL: The url of the posting on Ebay.

autoPay: Boolean. This field indicates if the seller requests immediate payment for the item. If true, immediate payment is required before the checkout process can begin. If false, immediate payment is not requested.

postalCode: Postal code of seller

returnsAccepted: Boolean. Will accept returns or not.

condition: Ebay has codes for condition such as the following (https://developer.ebay.com/Devzone/finding/CallRef/Enums/conditionIdList.html):

1000 - New. A brand-new, unused, unopened, unworn, undamaged item.

3000 - Used. An item that has been used previously. The item may have some signs of cosmetic wear, but is fully operational and functions as intended.

These are the two codes that pertain to this project.

topRatedListing: Indicates whether the item is Top Rated Plus item (https://pages.ebay.com/topratedplus/index.html). A top rated plus item:
- is listed by experienced sellers with highest buyer ratings;
- Sellers commit to shipping your items in a business day with tracking provided and offer at least a 14-day, money-back return policy;

pictureURLLarge: URL for item's picture url with size 400x400

pictureURLSuperSize: URL for item's picture url with size 800x800

shipping_cost: Cost that seller chose to add to total price for shipping purposes.

price_in_US: Price of the listing without shipping in US currency.

converted_price: Price of listing with shipping in US currency.

cost: fixed cost of the knife at the surplus store

profit: This column corresponds to the potential profit of buying a certain knife at the surplus store and then selling it on ebay. Total money recieved for the knife - total cost of the knife (cost at surplus store plus shipping plus overhead cost).

ROI: Return on Investment in percent of american dollars.

PictureURL: List. This field shows the URL to a full-size version of one image associated with the eBay listing. A PictureURL field is returned for each image in the eBay listing. At least one PictureURL field is always returned since every eBay listing must have at least one picture. Max value is 12 for the relevant item category.

Location: City, State location of listing

Country: country of listing

Everything below was information taking from the "itemSpecifics" call to the Shopping API which returned a list of item:value pairs that correspond to what a seller on Ebay is supposed to fill out when posting a listing to sell. For this project, this cooresponds to a list of aspects of each knife determined by each seller, including grammatical and spelling errors

Model: Brand related column. Each brand of knive has a line of models that they sell.

Country/Region of Manufacture: Country knife was manufactured.

Blade Material: Material of the knife blade

Blade Type: Shape of the blade. Most common blade typesL

Blade Edge: Is the edge serrated or something else?

Dexterity: Left handed or right handed or ambi.

Type: Item Specific value filled in by the seller about what type of knife they are selling, eg pocketknife or multitool

Color: Color of blade handle.

Number of Blades

Opening Mechanism: Assisted opening or something else?

Handle Material

Lock Type: Manual lock or something else?

Product Line: Brand specific column, similar to model.

Blade Range: The length of the blade acceptable range according to manufacturer

In [45]:
```
1 aspect_df = pd.read_csv('listed_data/listed_aspect_df.csv')
```

In [46]:
```
str_columns = ['Location', 'Country', 'Model',
               'Country/Region of Manufacture',
               'Blade Material', 'Blade Type',
               'Blade Edge', 'Dexterity',
               'Color', 'Number of Blades',
               'Opening Mechanism', 'Handle Material',
               'Lock Type', 'Blade Range']

for col in str_columns:
    aspect_df[col] = aspect_df[col].str.lower()

pattern = ".*,\s*([^\d,]+?)(?:\s*\d+)?$"
aspect_df['State_or_Province'] = aspect_df['Location'].str.extract(pattern)

aspect_df.head(10).transpose()
```

Out[46]:

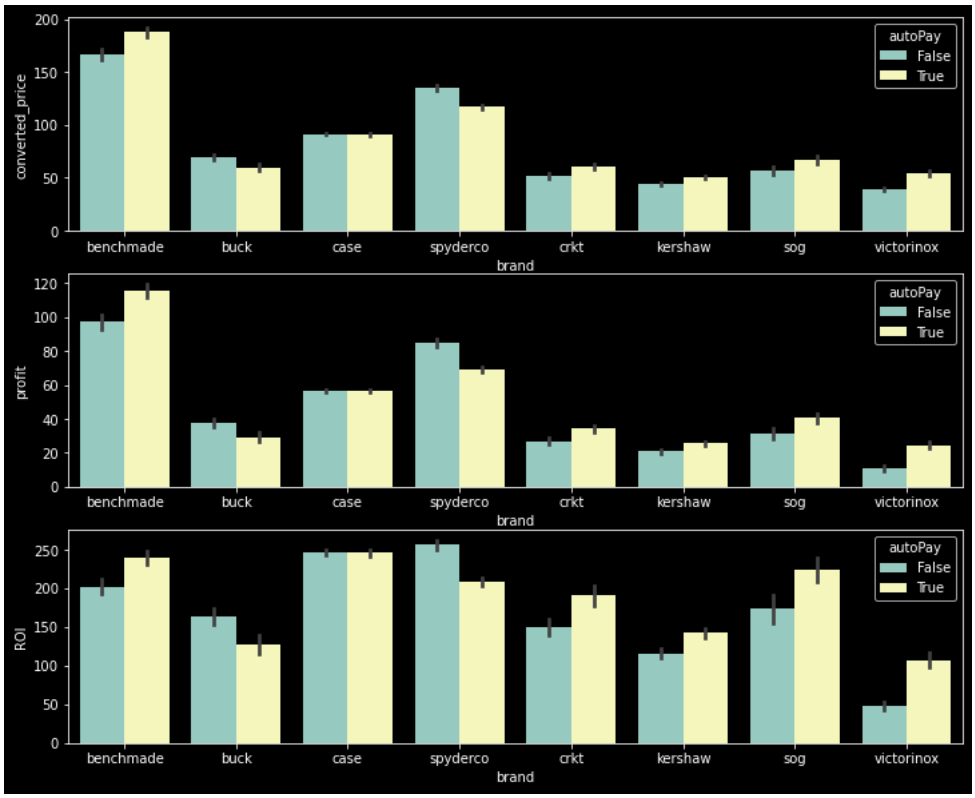
	0	1	
itemId	133142868285	143356945873	125491
title	benchmade pardue griptilian drop pt grip axs s...	benchmade pardue, mini griptilian, axs, hole 5...	benchmade 551-s30v steel drop-point blac
galleryURL	https://i.ebayimg.com/thumbs/images/g/xAoAAOSw...	https://i.ebayimg.com/thumbs/images/g/Te0AAOSw...	https://i.ebayimg.com/thumbs/images/g/e5lA
viewItemURL	https://www.ebay.com/itm/Benchmade-PARDUE-Grip...	https://www.ebay.com/itm/Benchmade-PARDUE-MINI...	https://www.ebay.com/itm/Benchmade-55
autoPay	True	True	
postalCode	020**	020**	
returnsAccepted	True	True	
condition	1000	1000	
topRatedListing	True	True	
pictureURLLarge	https://i.ebayimg.com/00/s/MTQ5OVgxNTY0/z/xAoA...	https://i.ebayimg.com/00/s/MTQ2OVgxNTY0/z/Te0A...	https://i.ebayimg.com/00/s/MTQ2Mlg1NjY=/
pictureURLSuperSize	https://i.ebayimg.com/00/s/MTQ5OVgxNTY0/z/xAoA...	https://i.ebayimg.com/00/s/MTQ2OVgxNTY0/z/Te0A...	https://i.ebayimg.com/00/s/MTQ2Mlg1NjY=/
shipping_cost	0	0	
price_in_US	106.25	117	
converted_price	106.25	117	
brand	benchmade	benchmade	ben
cost	45	45	
profit	44.4375	53.79	
ROI	92.5781	112.062	
PictureURL	['https://i.ebayimg.com/00/s/MTQ5OVgxNTY0/z/xA...	['https://i.ebayimg.com/00/s/MTQ2OVgxNTY0/z/Te...	['https://i.ebayimg.com/00/s/MTQ2Mlg1NjY
Location	canton, massachusetts	canton, massachusetts	
Country	us	us	
Blade Material	s-30v	stainless steel	stainle
Model	griptilian 551	mini griptilian 555	5
Opening Mechanism	manual	manual	
Number of Blades	1	1	
Handle Material	nylon	nylon	glass-fille
Blade Type	NaN	NaN	dr
Color	NaN	NaN	
Type	Pocketknife	Pocketknife	
Country/Region of Manufacture	NaN	united states	
Lock Type	NaN	axis	a
Blade Edge	NaN	plain	
Dexterity	NaN	ambidextrous	ambic
Blade Range	NaN	2.76 - 4in.	
State_or_Province	massachusetts	massachusetts	

In [47]:
```
aspect_df = apply_iqr_filter(aspect_df).copy()
```

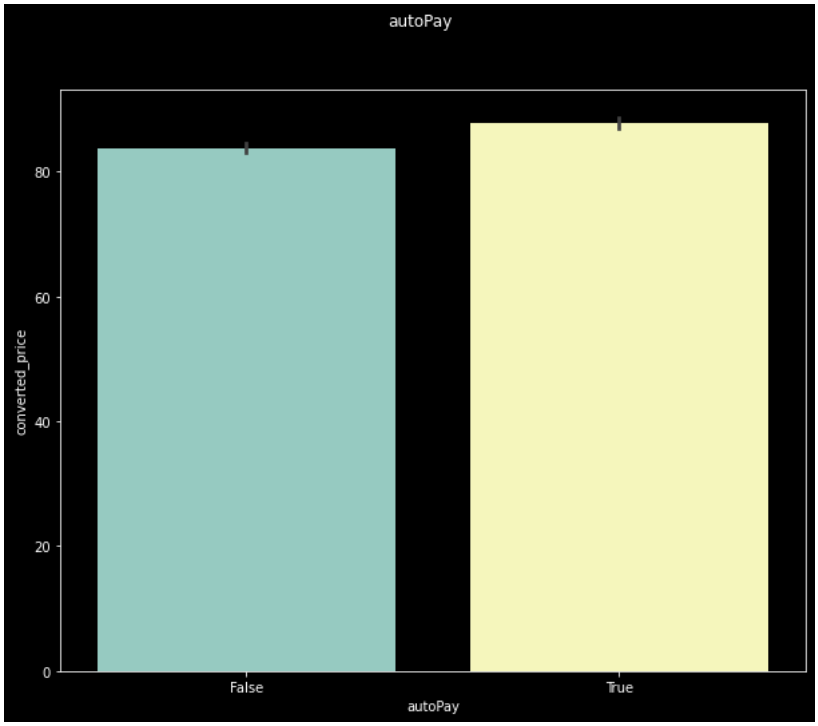
In [48]:
```
#drop uneeded columns
aspect_df.drop(['shipping_cost',
                'price_in_US','cost',
                'Type'], axis=1,
                inplace=True)
```

In [50]:
```
display(aspect_df['autoPay'].value_counts())

fig, axes  = plt.subplots(figsize=(12,10),nrows=3)
sns.barplot(x= 'brand', y='converted_price', ax=axes[0],hue='autoPay',data=aspect_df)
sns.barplot(x= 'brand', y='profit', ax=axes[1],hue='autoPay',data=aspect_df)
sns.barplot(x= 'brand', y='ROI', ax=axes[2],hue='autoPay',data=aspect_df)
plt.show();
```

```
False    19560
True     17106
Name: autoPay, dtype: int64
```



In [51]:
```
fig = plt.figure(figsize=(10,8))
fig.suptitle('autoPay')
sns.barplot(x= 'autoPay', y='converted_price', data=aspect_df)
plt.show();
```

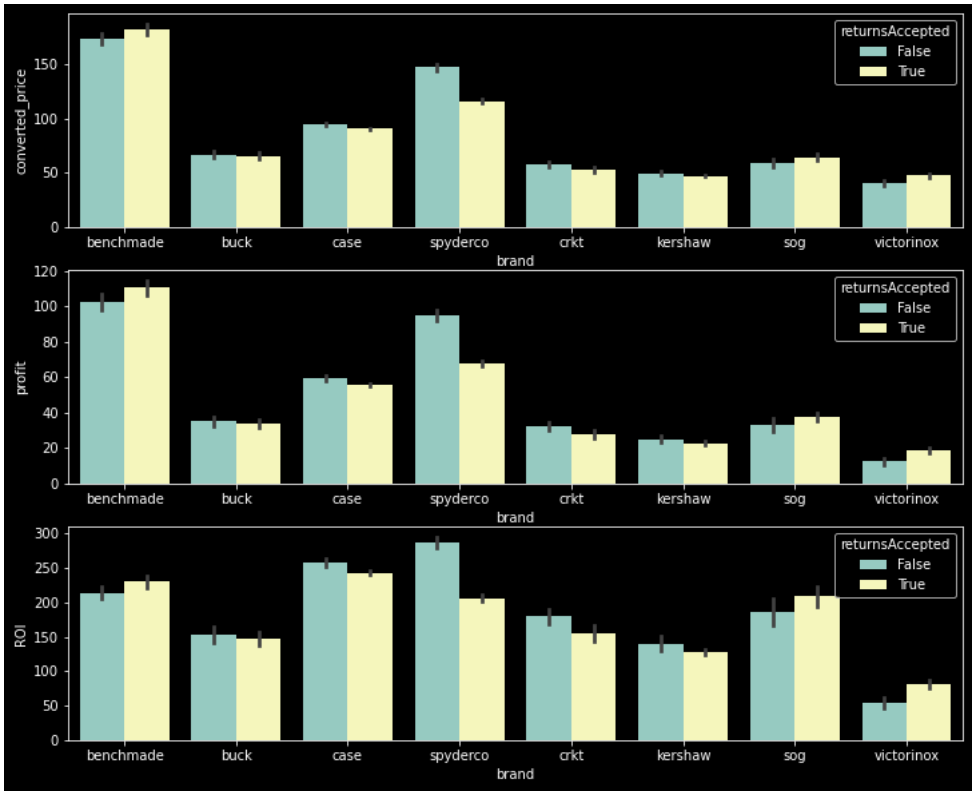


Returns Accepted

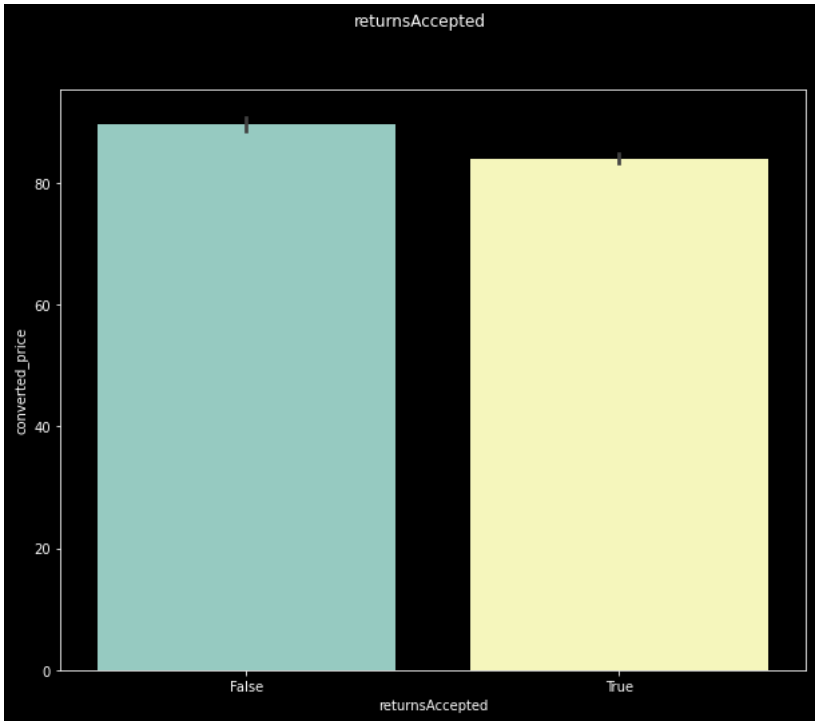
This boolean feature also appears to affect the price of listed factory made knives for sale. Overall, it would appear that sellers posting a knife that accept returns list the value of knives as slightly higher.

In [52]:
```
display(aspect_df['returnsAccepted'].value_counts())

fig, axes  = plt.subplots(figsize=(12,10),nrows=3)
sns.barplot(x= 'brand', y='converted_price', ax=axes[0],hue='returnsAccepted',data=aspect_df)
sns.barplot(x= 'brand', y='profit', ax=axes[1],hue='returnsAccepted',data=aspect_df)
sns.barplot(x= 'brand', y='ROI', ax=axes[2],hue='returnsAccepted',data=aspect_df)
plt.show();
```

```
True     25813
False    10853
Name: returnsAccepted, dtype: int64
```



In [53]:
```
fig = plt.figure(figsize=(10,8))
fig.suptitle('returnsAccepted')
sns.barplot(x= 'returnsAccepted', y='converted_price', data=aspect_df)
plt.show();
```



Barplots Exploring Knife Listed Price, Profits, and ROI by Brand

Condition

There is a clear differential between used and new knives. New knives are valued higher than used knives on ebay.

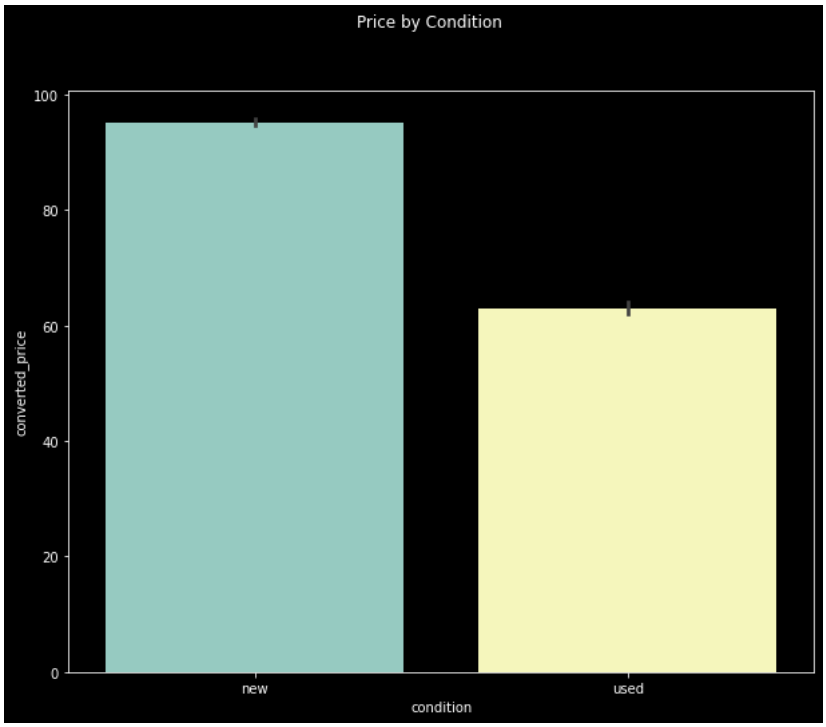
In [54]:
```
to_drop = aspect_df.loc[(aspect_df['condition'].isna()) |
                        (aspect_df['condition'] == 1500.00) |
                        (aspect_df['condition'] ==4000.0)].index
```

In [55]:
```
df_listed2 = aspect_df.drop(to_drop).copy()
```

In [56]:
```
df_listed2.condition.value_counts()
```

Out[56]:
```
1000.0    25758
3000.0    10906
Name: condition, dtype: int64
```

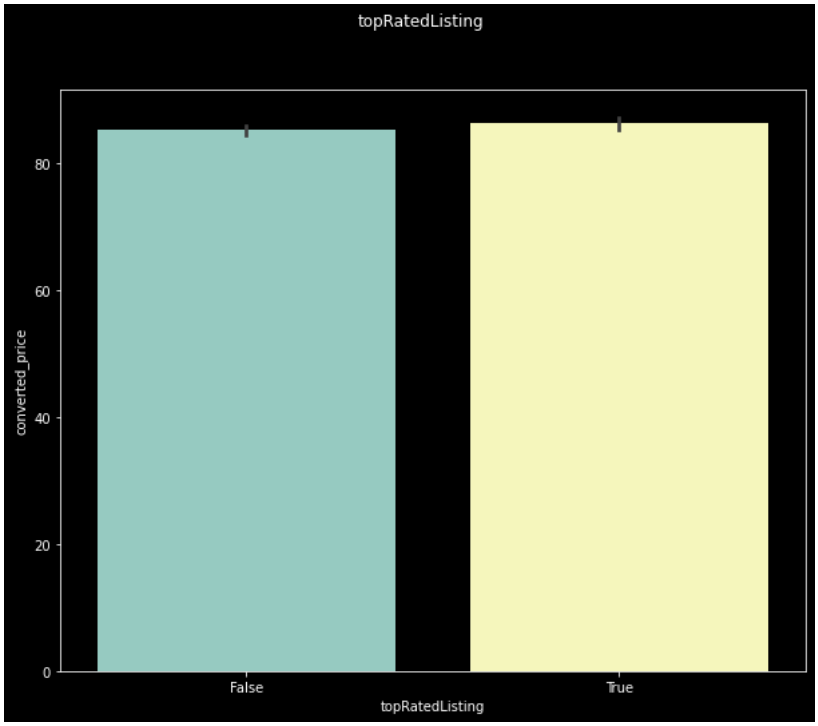
In [57]:
```
fig = plt.figure(figsize=(10,8))
fig.suptitle('Price by Condition')
sns.barplot(x= 'condition',
            y='converted_price',
            data=df_listed2)
plt.xticks([0,1], ['new', 'used'])

plt.show();
```



Top Rated Listing

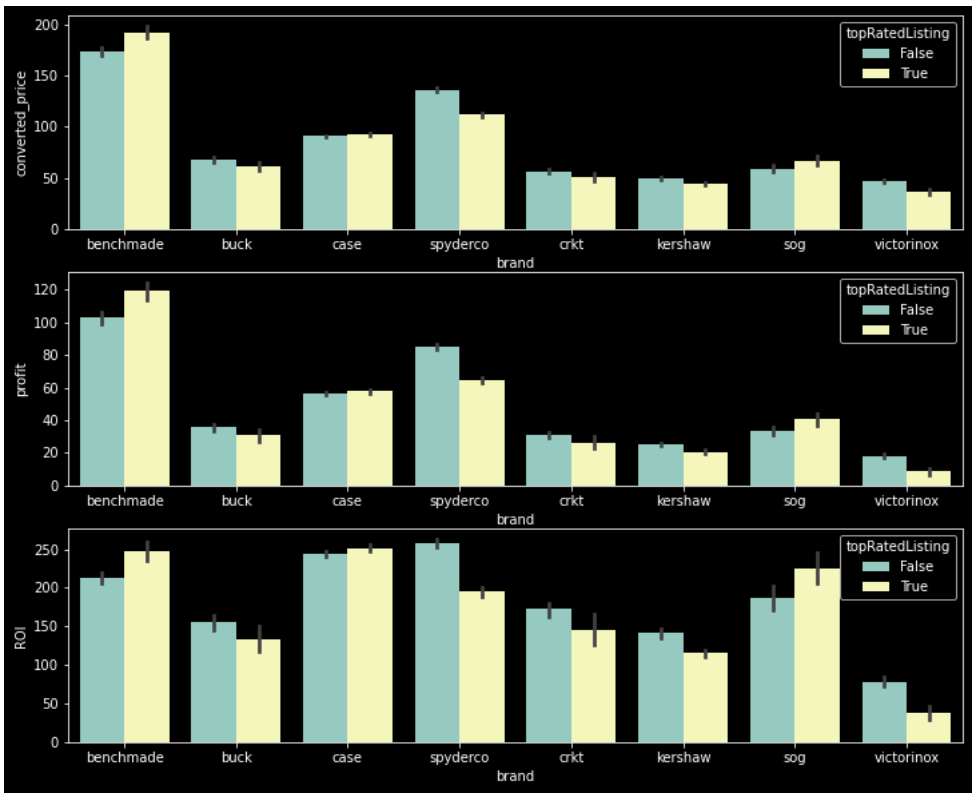
It seems that being Top Rated Listing might slightly increase the value of a knife sold on ebay. However, the difference is very slight.

In [58]:
```
fig = plt.figure(figsize=(10,8))
fig.suptitle('topRatedListing')
sns.barplot(x= 'topRatedListing', y='converted_price', data=aspect_df)
plt.show();
```



In [62]:
```
display(aspect_df['topRatedListing'].value_counts())

fig, axes  = plt.subplots(figsize=(12,10),nrows=3)
sns.barplot(x= 'brand', y='converted_price', ax=axes[0],hue='topRatedListing',data=aspect_df)
sns.barplot(x= 'brand', y='profit', ax=axes[1],hue='topRatedListing',data=aspect_df)
sns.barplot(x= 'brand', y='ROI', ax=axes[2],hue='topRatedListing',data=aspect_df)
plt.show();
```

```
False    23602
True     13064
Name: topRatedListing, dtype: int64
```



Large Cardinality Categorical Data

Location

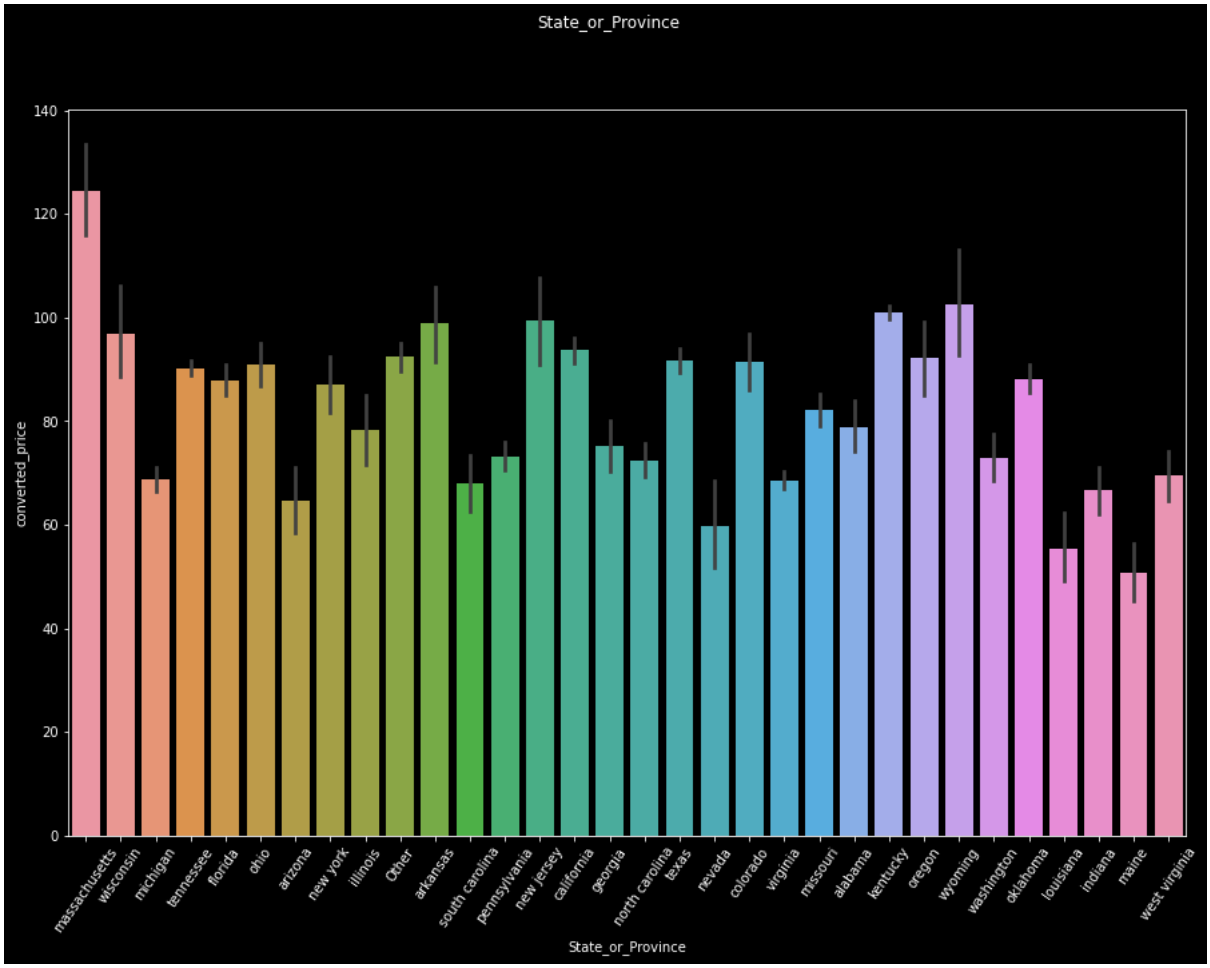
In [63]:
```
transformed_loc = cardinality_threshold(aspect_df['Location'],
                                        threshold=0.7,
                                        return_categories_list=False)
transformed_loc.value_counts()
```

Out[63]:
```
Other                          11000
knoxville, tennessee            3018
prestonsburg, kentucky          1092
crestwood, kentucky             1011
coeburn, virginia                905
                                ...
north richland hills, texas       33
radcliff, kentucky                32
waynesville, ohio                 32
glasgow, kentucky                 32
crest hill, illinois              32
Name: Location, Length: 192, dtype: int64
```

In [64]:
```
transformed_state = cardinality_threshold(aspect_df['State_or_Province'],
                                        threshold=0.93,
                                        return_categories_list=False)
transformed_state.value_counts()
```

Out[64]:
```
kentucky          4268
tennessee         4157
california        2779
virginia          2670
texas             2573
Other             2356
florida           1545
michigan          1514
pennsylvania      1444
missouri          1189
north carolina    1166
ohio               824
oklahoma           811
washington         709
colorado           701
georgia            653
indiana            649
new york           581
alabama            481
south carolina     436
illinois           380
wyoming            364
arizona            353
maine              344
oregon             297
west virginia      296
wisconsin          279
new jersey         276
arkansas           271
massachusetts      269
nevada             254
louisiana          246
Name: State_or_Province, dtype: int64
```

In [65]:
```
df_transformed = aspect_df.copy()
df_transformed['State_or_Province'] = transformed_state
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('State_or_Province')
sns.barplot(x= 'State_or_Province', y='converted_price', data=df_transformed)
plt.xticks(rotation=55)
plt.show();
```



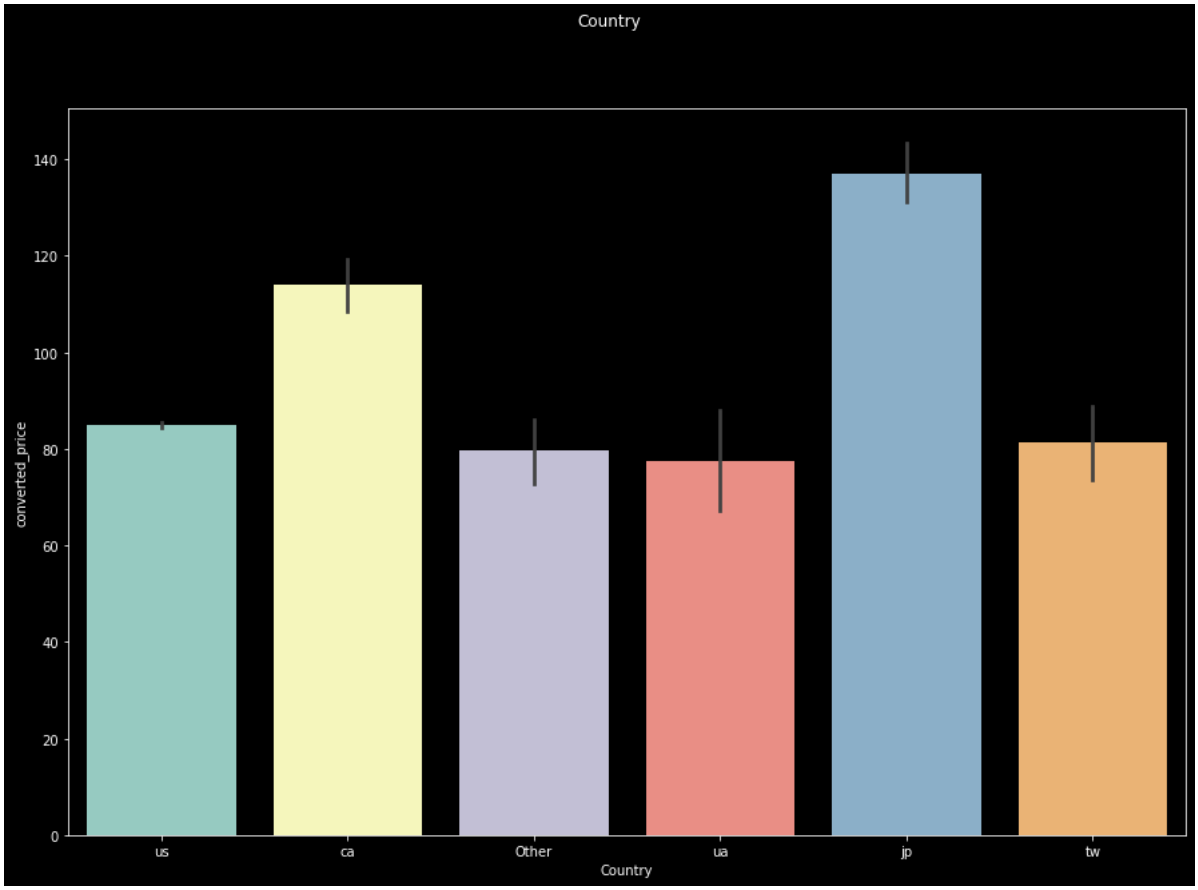
In [66]:
```
transformed_country = cardinality_threshold(aspect_df['Country'],
                                            threshold=0.992,
                                            return_categories_list=False)
transformed_country.value_counts()
```

Out[66]:
```
us       35381
ca         493
jp         318
Other      207
ua         157
tw         110
Name: Country, dtype: int64
```

Country

Japan knives appear to be valued higher in general when compared to other countries, as well as Canada, but not as intenesly as Japan.

In [67]:
```
df_transformed = aspect_df.copy()
df_transformed['Country'] = transformed_country
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('Country')
sns.barplot(x= 'Country', y='converted_price', data=df_transformed)
plt.show();
```



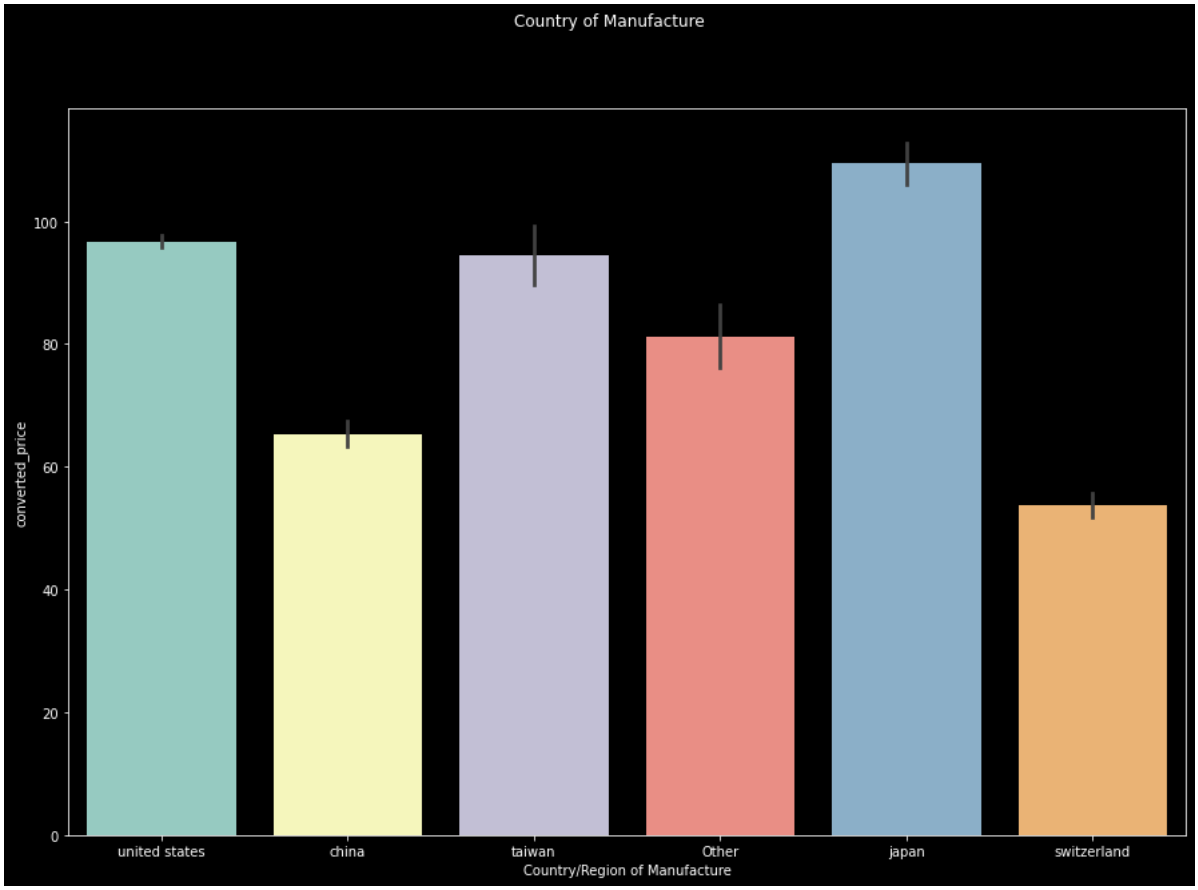
In [68]:
```
transformed_rom = cardinality_threshold(aspect_df['Country/Region of Manufacture'],
                                        threshold=0.98,
                                        return_categories_list=False)
transformed_rom.value_counts()
```

Out[68]:
```
united states    13073
switzerland       2426
china             2371
japan             1664
taiwan             599
Other              442
Name: Country/Region of Manufacture, dtype: int64
```

Country of Manufacture

Japanese manufactured knives appear to be more valuable than other manufactured regions. China or swiss knives (mainly victorinox) seem to skew lower in value as compared to manufactured knives in other countries.

In [69]:
```
df_transformed = aspect_df.copy()
df_transformed['Country/Region of Manufacture'] = transformed_rom
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('Country of Manufacture')
sns.barplot(x= 'Country/Region of Manufacture', y='converted_price', data=df_transformed)
plt.show();
```



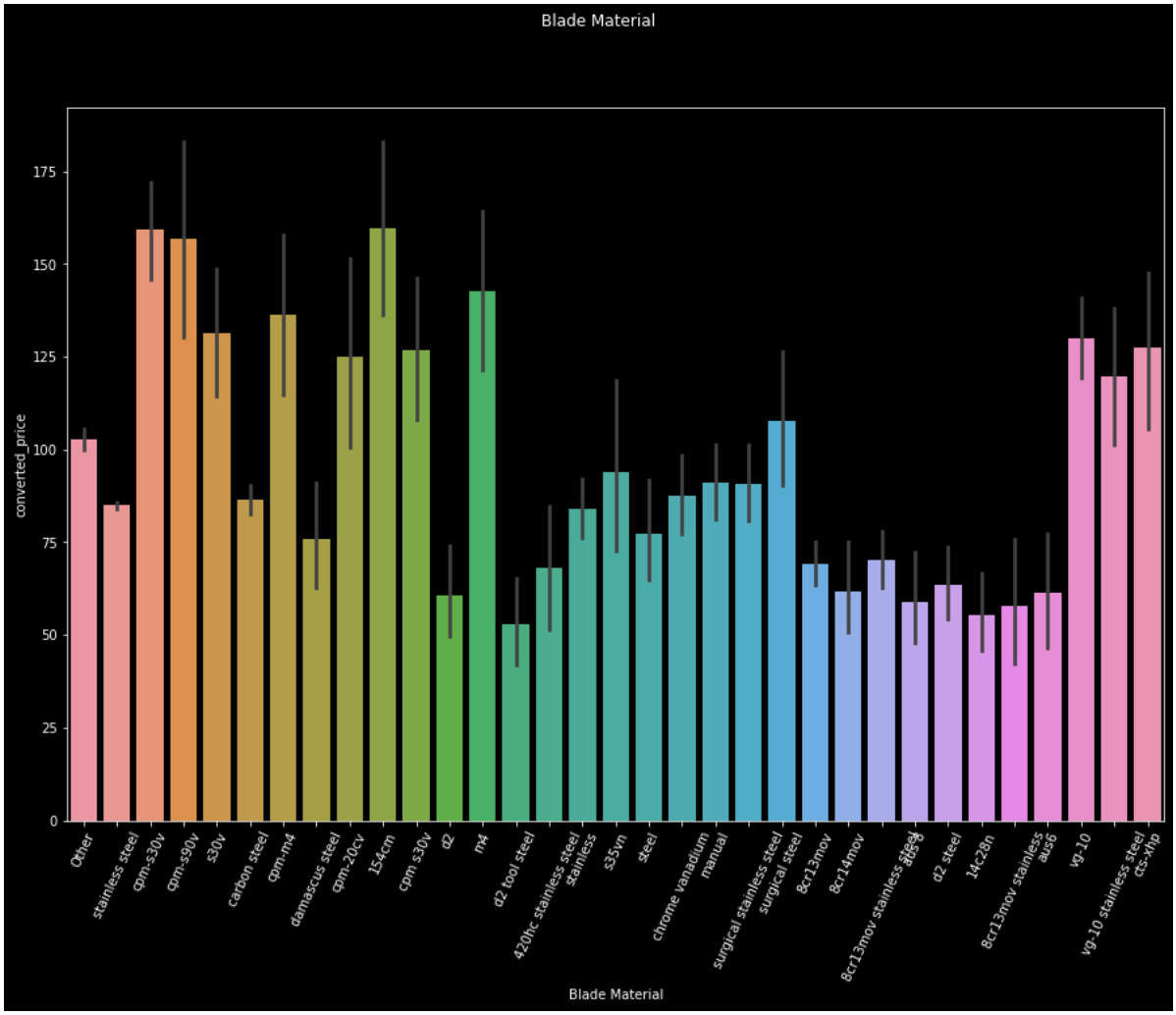
Blade Material

There is clear variance between knives of different material. Most blades are some form of stainless steel. Grades of steel can be assumed to affect prices as knives as some materials are more sturdy and sharpen better than other materials. Some materials are also more expensive than others.

In [70]:
```
transformed_blm = cardinality_threshold(aspect_df['Blade Material'],
                                        threshold=0.94,
                                        return_categories_list=False)
transformed_blm.value_counts()
```

Out[70]:
```
stainless steel             16198
Other                        2174
carbon steel                  793
8cr13mov                      309
vg-10                         162
8cr13mov stainless steel      151
stainless                     139
cpm-s30v                       98
chrome vanadium                85
manual                         82
damascus steel                 81
steel                          80
s30v                           79
surgical stainless steel       74
d2                             58
d2 steel                       57
420hc stainless steel          51
cpm s30v                       50
vg-10 stainless steel          41
14c28n                         38
cts-xhp                        38
8cr13mov stainless             38
cpm-m4                         35
d2 tool steel                  35
m4                             35
cpm-20cv                       32
aus 8                          31
154cm                          30
s35vn                          29
aus6                           28
8cr14mov                       28
cpm-s90v                       28
surgical steel                 27
Name: Blade Material, dtype: int64
```

In [71]:
```
df_transformed = aspect_df.copy()
df_transformed['Blade Material'] = transformed_blm
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('Blade Material')
sns.barplot(x= 'Blade Material', y='converted_price', data=df_transformed)
plt.xticks(rotation=65)
plt.show();
```



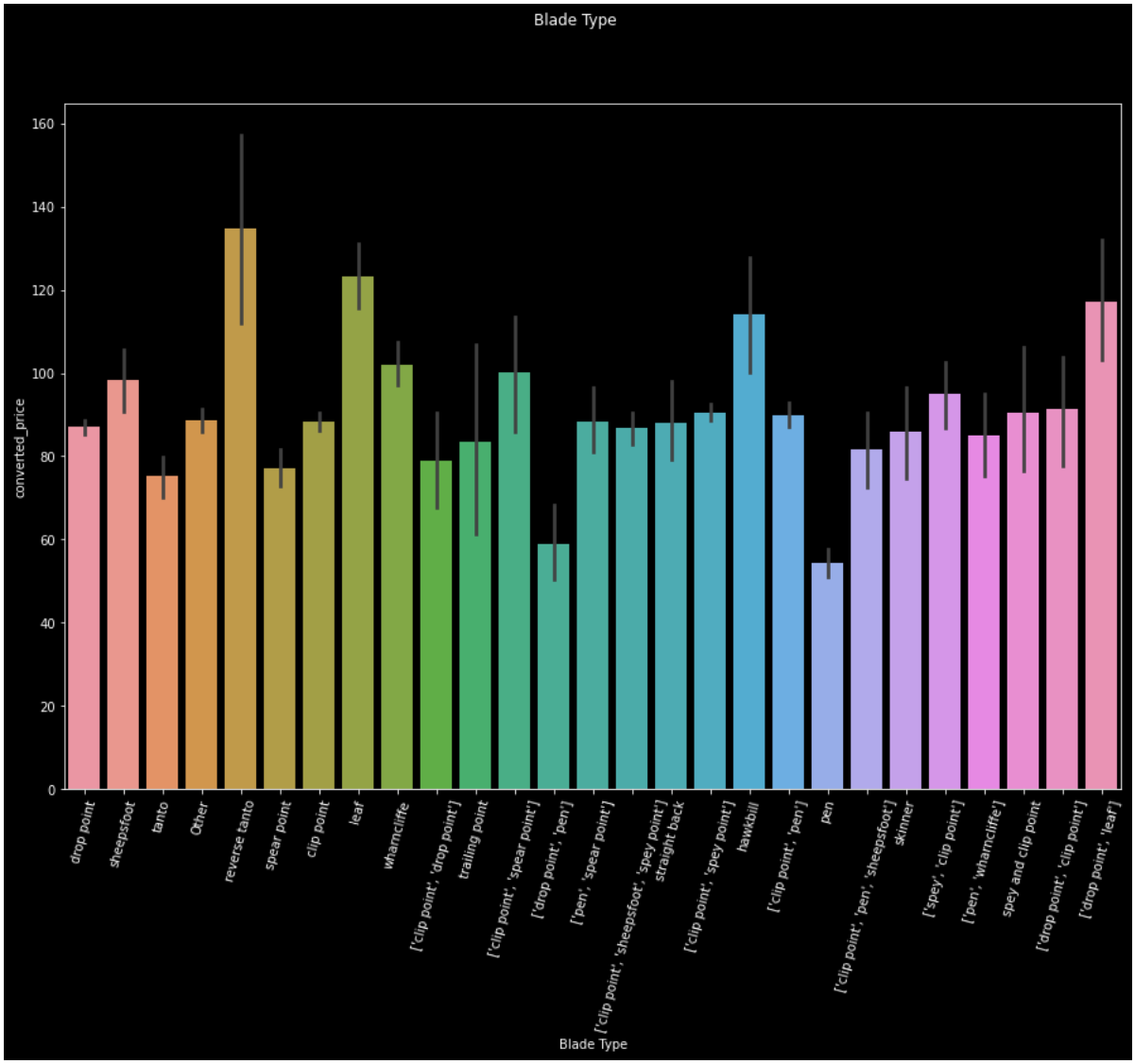
Blade Type

There is clear variance between knives of different blade type. Most blades are drop, clip, spear, or tanto knives. Knife collectors sort different blade types for different reasons and this can have an affect on the price,

In [72]:
```
transformed_blt = cardinality_threshold(aspect_df['Blade Type'],
                                        threshold=0.95,
                                        return_categories_list=False)
transformed_blt.value_counts()
```

Out[72]:
```
drop point                                  4826
clip point                                  2980
Other                                       1815
['clip point', 'spey point']                1730
['clip point', 'pen']                        779
pen                                          778
spear point                                  639
tanto                                        615
wharncliffe                                  537
['clip point', 'sheepsfoot', 'spey point']   523
leaf                                         318
sheepsfoot                                   315
['pen', 'spear point']                       145
['spey', 'clip point']                       133
straight back                                130
['drop point', 'pen']                        122
hawkbill                                      91
['drop point', 'leaf']                        86
['drop point', 'clip point']                  86
skinner                                       84
['clip point', 'pen', 'sheepsfoot']           78
['clip point', 'drop point']                  73
['pen', 'wharncliffe']                        70
reverse tanto                                 60
['clip point', 'spear point']                 48
spey and clip point                           43
trailing point                                37
Name: Blade Type, dtype: int64
```

In [73]:
```
df_transformed = aspect_df.copy()
df_transformed['Blade Type'] = transformed_blt
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('Blade Type')
sns.barplot(x= 'Blade Type', y='converted_price', data=df_transformed)
plt.xticks(rotation=75)
plt.show();
```



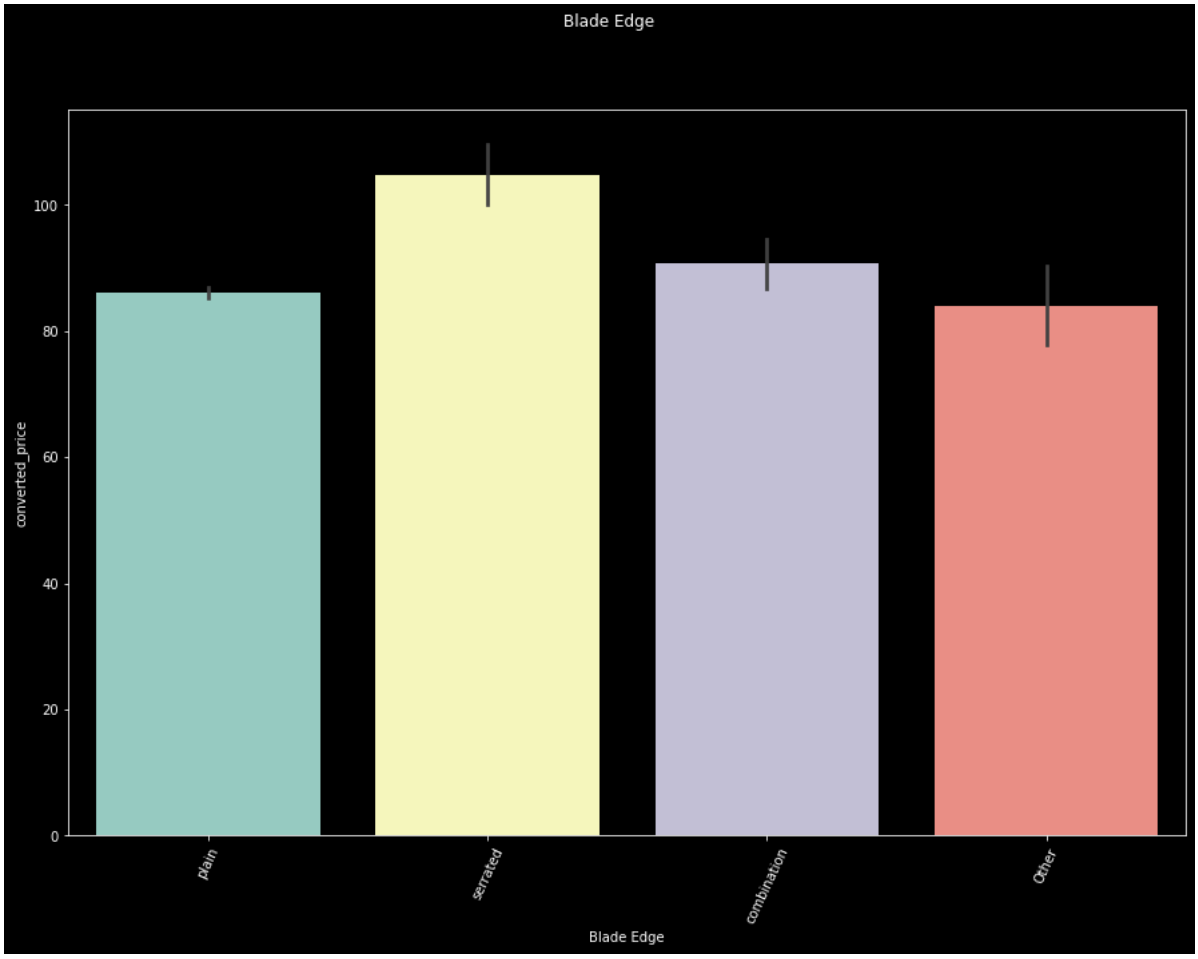
Blade Edge

Serrated knives or combination plain-and-serrated edge knives appear to be listed at a higher value than just plain edged knives.

In [74]:
```
transformed_ble = cardinality_threshold(aspect_df['Blade Edge'],
                                        threshold=0.99,
                                        return_categories_list=False)
transformed_ble.value_counts()
```

Out[74]:
```
plain          17433
combination     1164
serrated         766
Other            310
Name: Blade Edge, dtype: int64
```

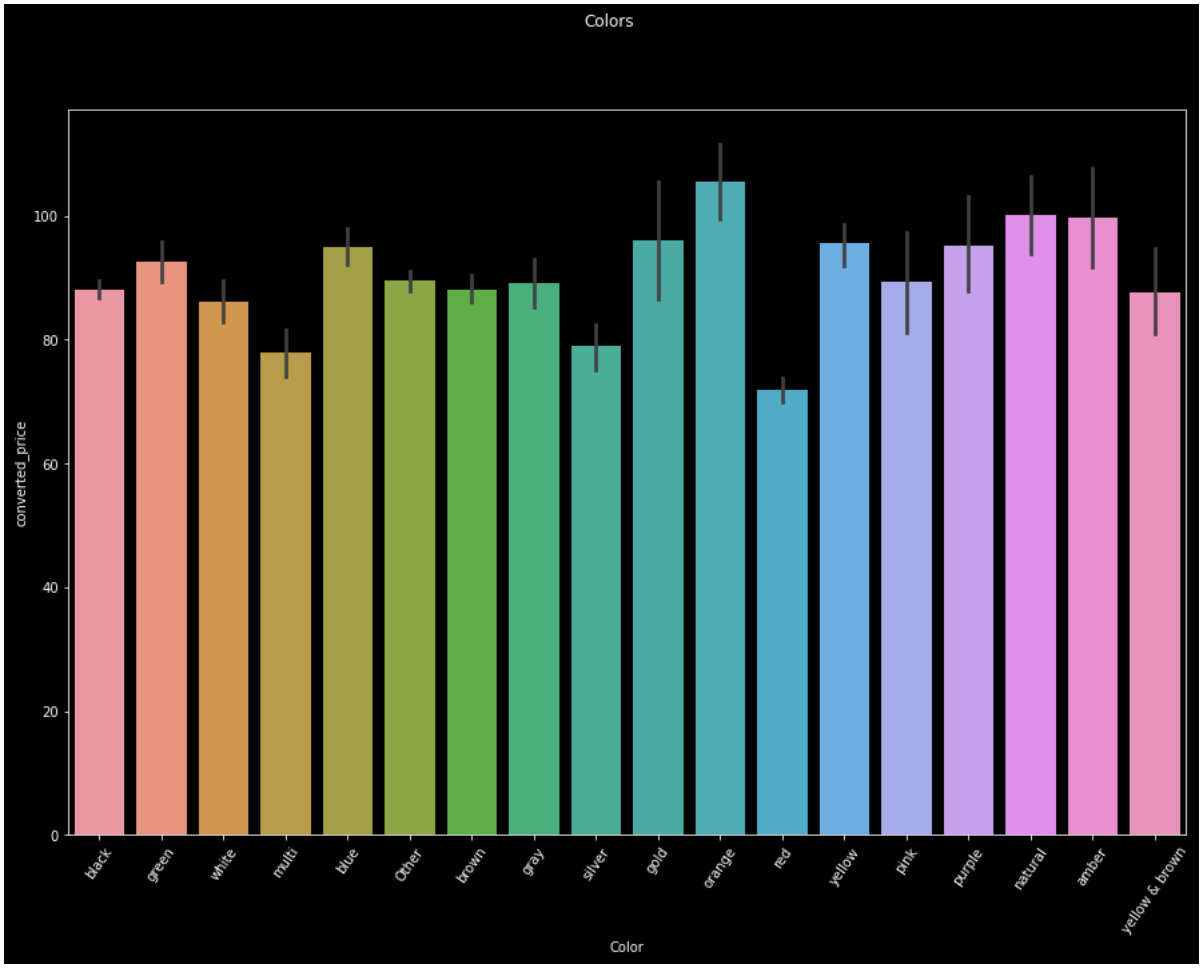
In [75]:
```
df_transformed = aspect_df.copy()
df_transformed['Blade Edge'] = transformed_ble
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('Blade Edge')
sns.barplot(x= 'Blade Edge', y='converted_price', data=df_transformed)
plt.xticks(rotation=65)
plt.show();
```



In [76]:
```
transformed_color = cardinality_threshold(aspect_df['Color'],
                                          threshold=0.88,
                                          return_categories_list=False)
transformed_color.value_counts()
```

Out[76]:
```
black            7129
Other            4319
red              2797
brown            2213
blue             1697
green            1304
silver           1126
gray             1097
multi             858
yellow            854
white             748
orange            469
purple            255
natural           222
pink              215
gold              172
amber             157
yellow & brown    144
Name: Color, dtype: int64
```

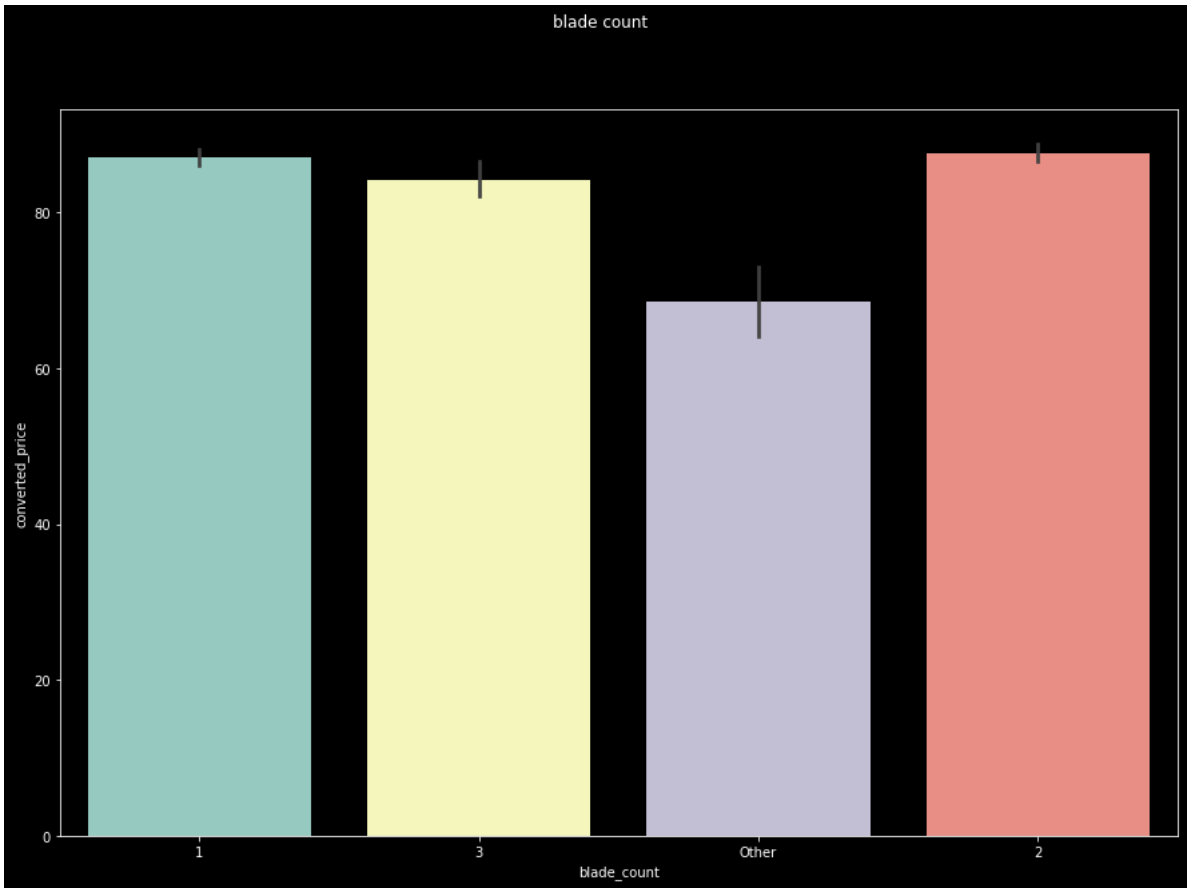
In [77]:
```
df_transformed = aspect_df.copy()
df_transformed['Color'] = transformed_color
# df_sorted = df_transformed.sort_values('converted_price')

fig = plt.figure(figsize=(15,10))
fig.suptitle('Colors')
sns.barplot(x= 'Color', y='converted_price', data=df_transformed)
plt.xticks(rotation=55)
plt.show();
```



Number of Blades

There seems to be little variance between knife value when it comes to having 1,2, or 3 blades. However, increasing the number of blades to 4 or more seems to have a negative impact on the value.

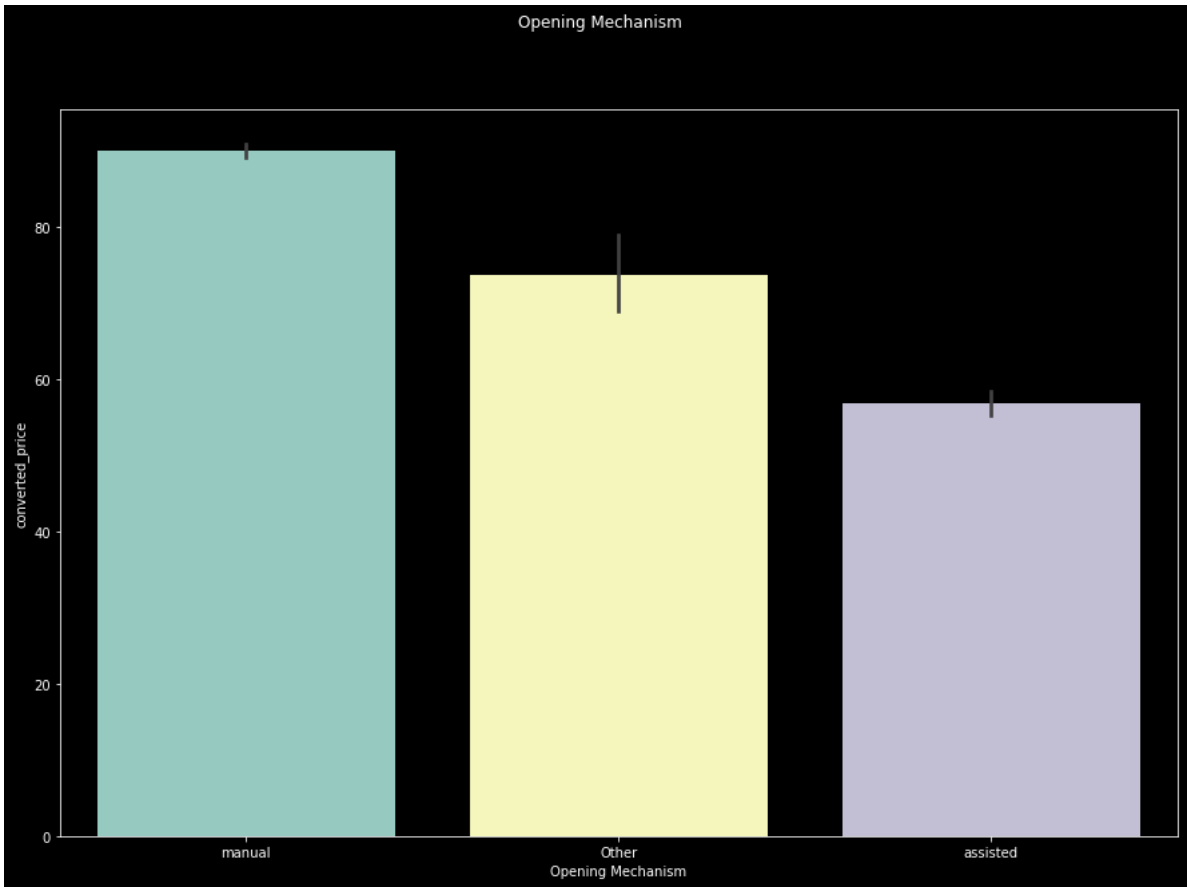
In [78]:
```
transformed_blade_count = cardinality_threshold(aspect_df['Number of Blades'],
                                                threshold=0.97,
                                                return_categories_list=False)
transformed_blade_count.value_counts()
df_transformed = aspect_df.copy()
df_transformed['blade_count'] = transformed_blade_count


fig = plt.figure(figsize=(15,10))
fig.suptitle('blade count')
sns.barplot(x= 'blade_count', y='converted_price', data=df_transformed)
plt.show();
```



Opening Mechanism

The barplot below would suggest that overall, manual opening knives are listed at a higher value than assisted open knives on ebay.

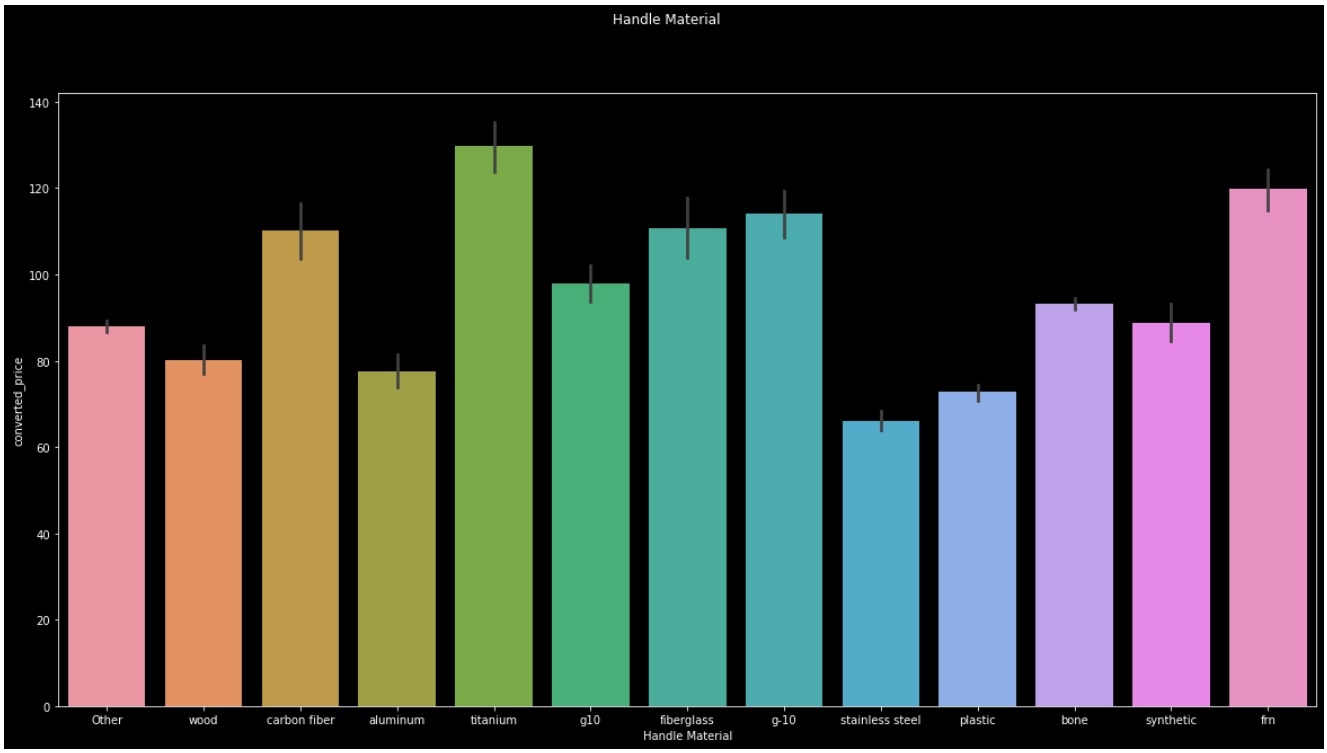
In [79]:
```
transformed_OM = cardinality_threshold(aspect_df['Opening Mechanism'],
                                       threshold=0.95,
                                       return_categories_list=False)

df_transformed = aspect_df.copy()
df_transformed['Opening Mechanism'] = transformed_OM


fig = plt.figure(figsize=(15,10))
fig.suptitle('Opening Mechanism')
sns.barplot(x= 'Opening Mechanism', y='converted_price', data=df_transformed)
plt.show();
```



Handle Material

Different handle materials cost different amounts to produce and there seems to be a perception of different value when it comes to different handle materials. The barplot below suggests variance in listed prices for knives based on handle material.

In [80]:
```
transformed_HM = cardinality_threshold(aspect_df['Handle Material'],
                                       threshold=0.8,
                                       return_categories_list=False)

df_transformed = aspect_df.copy()
df_transformed['Handle Material'] = transformed_HM


fig = plt.figure(figsize=(20,10))
fig.suptitle('Handle Material')
sns.barplot(x= 'Handle Material', y='converted_price', data=df_transformed)
plt.show();
```



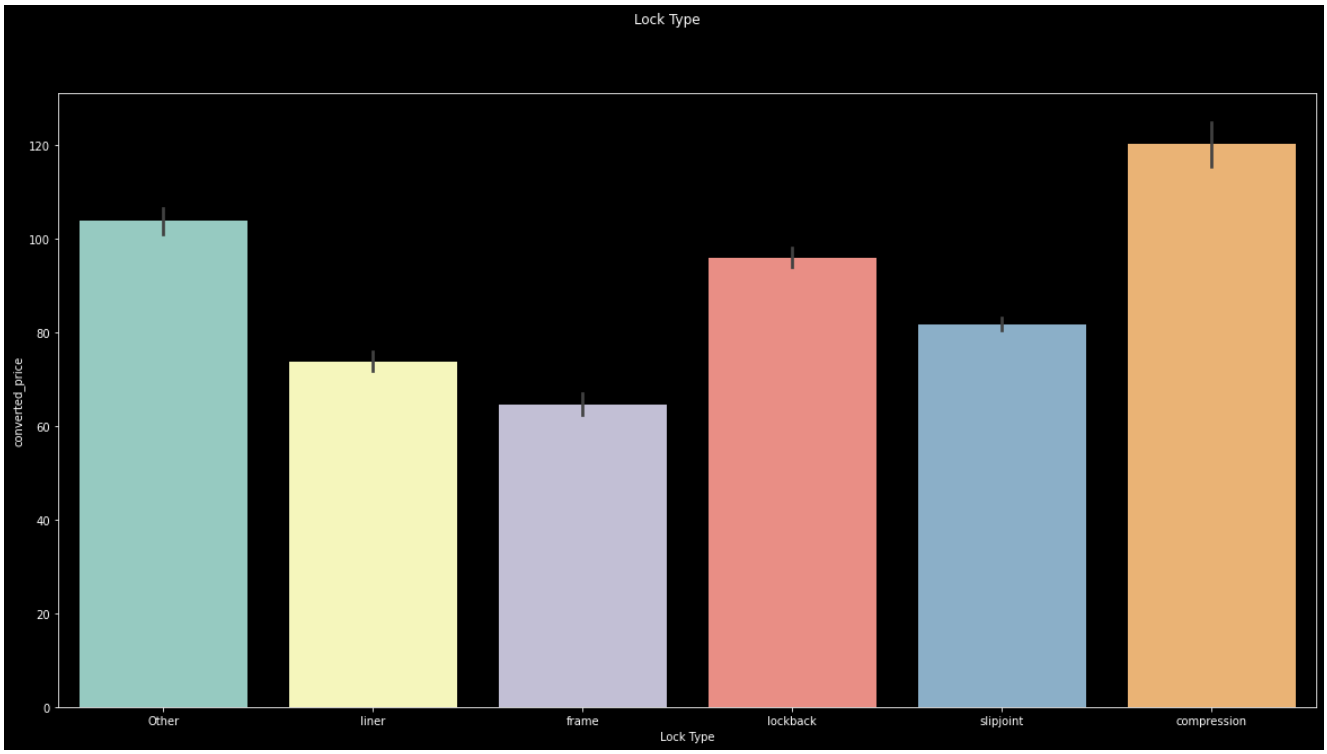
In [81]:
```
df_listed.columns
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-81-724eeb96ad5c> in <module>
----> 1 df_listed.columns

NameError: name 'df_listed' is not defined
```

Locktype

Locktype appears to have an affect on the price of knives.

In [82]:
```
transformed_LT = cardinality_threshold(aspect_df['Lock Type'],
                                       threshold=0.93,
                                       return_categories_list=False)


df_transformed = aspect_df.copy()
df_transformed['Lock Type'] = transformed_LT


fig = plt.figure(figsize=(20,10))
fig.suptitle('Lock Type')
sns.barplot(x= 'Lock Type', y='converted_price', data=df_transformed)
plt.show();
```



Noisy blade range data makes it hard to intpret the barplot for the transformed data.

In [83]:

```
transformed_BR = cardinality_threshold(aspect_df['Blade Range'],
                                       threshold=0.97,
                                       return_categories_list=False)


df_transformed = aspect_df.copy()
df_transformed['Blade Range'] = transformed_BR


fig = plt.figure(figsize=(20,10))
fig.suptitle('Blade Range')
sns.barplot(x= 'Blade Range', y='ROI', data=df_transformed)
plt.show();
```