DataInsight Corp.

The company is interested in optimizing its business operations, improving marketing strategies, and making informed decisions based on available data. They also want to find additional products to pursue and would like to know which type of product would work best.

Dataset

```
print(etsy_data.shape)

(1000, 47)
```

This dataset features **1,000** etsy products in the office gift niche. It has 47 columns and contains the basic listing information as **product name, shop name**, and **link** and more advanced analytics as **monthly sales**, **price, amount of reviews** and **listing age**.

I want to analyze these listings and find which price point and category sell the best and are good to pursue for our business. I'd also like to see if any additional features play a role in sales volume such as personalizable, where it's shipped from and if it auto-renews.

print(etsy_data.info())

```
0   product_name                  1000 non-null   object
1   product_link                  1000 non-null   object
2   shop_name                     1000 non-null   object
3   shop_link                     1000 non-null   object
4   price                         1000 non-null   int64
5   est_mo_sales                  1000 non-null   int64
6   est_mo_revenue                1000 non-null   int64
7   est_total_sales               1000 non-null   int64
8   reviews                       1000 non-null   int64
9   listing_age                   1000 non-null   object
10  favorites                     1000 non-null   int64
11  avg_reviews                   1000 non-null   int64
12  views                         1000 non-null   int64
13  category                      992 non-null    object
14  tags_used                     0 non-null      float64
15  auto_renews                   1000 non-null   bool
16  is_customizable               1000 non-null   bool
17  is_personalizable             1000 non-null   bool
18  description_character_count   0 non-null      float64
19  has_variations                1000 non-null   bool
20  is_supply                     1000 non-null   bool
21  minimum_processing            979 non-null    float64
22  placement_of_listing_in_shop  1000 non-null   int64
23  shipped_from                  986 non-null    object
24  shop_age                      1000 non-null   int64
25  visibility_score              1000 non-null   int64
26  conversion_rate               1000 non-null   float64
27  shop_digital_listing_count    1000 non-null   int64
28  title_character               1000 non-null   int64
29  when_made                     1000 non-null   object
30  who_made                      1000 non-null   object
31  total_shop_sales              1000 non-null   int64
```

Most data types seem to fit for that class.

- change listing age to integer it is an object now.
- add an additional row if it’s a gift box/ set or single product will extract it from the title.
- Some missing values in category, min processing and shipped from columns.
- Tags used and description count columns are completely empty so will drop those
- Would also drop columns I don’t need for analytics as the product link
- Will also probably drop the 13 tags columns for now.
- Need to decide which metric to use to see if the product is doing well. Do monthly sales or revenue or rather conversion rate or visibility score.

Semi Structured Data

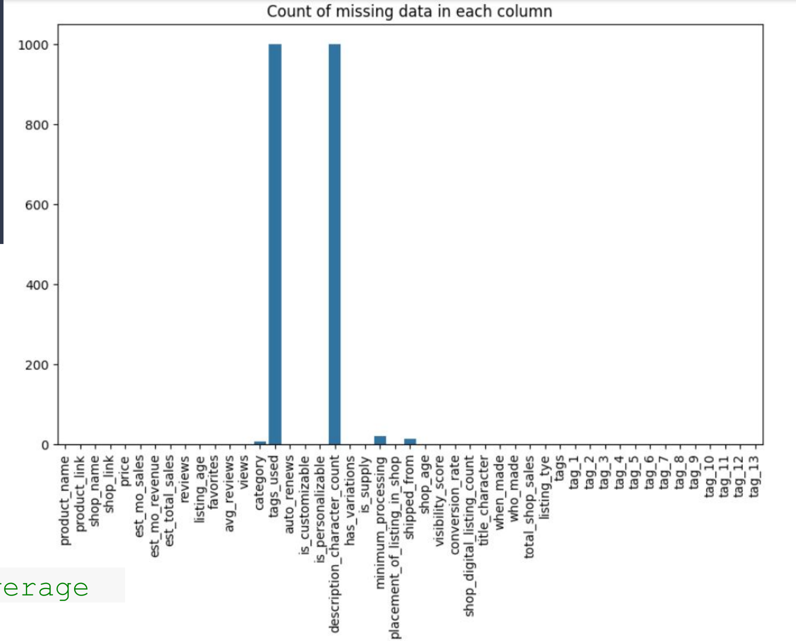
```
#add column if is a gift box from product name using semi structured data
as we can't do gift boxes now
# Create a column to identify rows containing the word 'box' or 'basket'
in any word in the string
etsy_data['gift box'] =
etsy_data['product_name'].str.contains('box|basket', case=False)
print((etsy_data['gift box']==True).sum()) # 153 are gift boxes remove
those
# filter to keep rows where the value does not contain 'box or basket'
#do it later before running models as first want to see overall stats
```

Unstructured Data

Would gather reviews left by customers on the products.

- Analyze sentiment to understand customer satisfaction levels.
- Identify common themes or issues mentioned in reviews to address any product shortcomings.
- Monitor trends in customer preferences and identify emerging needs or preferences.

Missing Values



Removed null values and columns not using.

```
change na in category to unknown and minimum processing to average
min_proc_mean = etsy_data[ 'minimum_processing' ].mean()
etsy_data[ 'minimum_processing' ] = etsy_data[ 'minimum_processing' ].fillna(min_proc_mean)
#Drop tags_used and description_character_count as empty column all are na
etsy_data = etsy_data.drop([ 'tags_used' , 'description_character_count' ], axis = 1)
#Drop all 13 tag columns and first 4 columns name, link , shop name , and shop link
etsy_data = etsy_data.drop([ 'product_name' , 'product_link' , 'shop_name' , 'shop_link' , 'tags', 'tag_1' ,
'tag_2' ,
       'tag_3', 'tag_4', 'tag_5', 'tag_6', 'tag_7', 'tag_8', 'tag_9', 'tag_10',
       'tag_11', 'tag_12', 'tag_13','shop_digital_listing_count' ,'placement_of_listing_in_shop' ], axis =
1)
```

Changed listing age type

```
#remove the mo from listing age column
etsy_data['listing_age'] = etsy_data['listing_age'].str.extract('(\d+)', expand=False)
#convert listing age to integer
etsy_data['listing_age'] = etsy_data['listing_age'].astype(int)
#change column name to age_months
etsy_data.rename(columns = {'listing_age':'listing_age_months'}, inplace=True)
#validate that changed to integer
print(etsy_data['listing_age_months'].dtype)
```

Extracted mo. after each number

Changed to integer

Renamed column to listing_age_months

etsy_data.describe()

- avg price is **$32**
- avg monthly sales are **63** sales.
- average total sales are around **$1200**
- avg amount of reviews are **240**
- **1,476** favorites on avg
- avg views per product are **35k**
- avg shop age is **60** months
- avg conversion rate is around **4**

	price	est_mo_sales	est_mo_revenue	est_total_sales	reviews	\
count	1000.00000	1000.000000	1000.000000	1000.0000	1000.000000	
mean	31.87400	63.419000	1504.655000	1226.0360	239.041000	
std	23.55689	106.636649	2784.942719	2124.7364	380.518346	
min	2.00000	3.000000	396.000000	9.0000	0.000000	
25%	16.00000	20.000000	535.500000	231.0000	48.000000	
50%	25.00000	35.000000	751.000000	589.0000	116.000000	
75%	42.00000	67.000000	1397.500000	1445.2500	293.000000	
max	220.00000	2003.000000	52052.000000	24324.0000	4682.000000	

	listing_age_months	favorites	avg_reviews	views	\
count	1000.00000	1000.000000	1000.000000	1000.000000	
mean	21.09400	1476.433000	12.768000	35532.265000	
std	19.35339	2707.377234	19.655241	57133.348099	
min	1.00000	0.000000	0.000000	0.000000	
25%	7.00000	218.000000	4.000000	6863.000000	
50%	16.00000	590.500000	7.000000	18262.500000	
75%	29.00000	1631.000000	14.000000	40066.000000	
max	138.00000	37442.000000	297.000000	622499.000000	

	minimum_processing	shop_age	visibility_score	conversion_rate	\
count	1000.000000	1000.000000	1000.000000	1000.000000	
mean	1.958121	60.757000	94.274000	4.140240	
std	2.149396	41.280213	18.285165	4.235549	
min	1.000000	2.000000	0.000000	0.000000	
25%	1.000000	30.000000	100.000000	2.137500	
50%	1.000000	48.000000	100.000000	3.245000	
75%	2.000000	88.000000	100.000000	4.910000	
max	20.000000	200.000000	100.000000	52.280000	

	title_character	total_shop_sales
count	1000.000000	1.000000e+03
mean	129.735000	6.423644e+04
std	18.624383	1.622795e+05
min	19.000000	7.700000e+01
25%	130.000000	5.977000e+03
50%	136.000000	2.115400e+04
75%	139.000000	6.581900e+04
max	150.000000	1.811687e+06

Viewing outliers and basic shape using pairplots

```
#do separate dataframe with only all success metrics
etsy_success_metrics = etsy_data[['est_mo_sales',
'est_mo_revenue', 'est_total_sales', 'reviews',
        'favorites', 'avg_reviews', 'views',
       'visibility_score', 'conversion_rate',
'total_shop_sales']]

from scipy.stats import pearsonr
#function to plot correlation coefficient
def corrfunc(x, y, ax=None, **kws):
    """Plot the correlation coefficient in the top
left hand corner of a plot."""
    r, _ = pearsonr(x, y)
    ax = ax or plt.gca()
    ax.annotate(f'ρ = {r:.2f}', xy=(.1, .9),
xycoords=ax.transAxes)

#pairplot with success metrics
g = sns.pairplot(etsy_success_metrics,corner=True)
#show correlation score for each plot
g.map_lower(corrfunc)
plt.show()
```

Outliers = products making more than 1,000 sales per month.

Will check out top outliers separately to see if can find what makes them sell so much more than rest.

Two products make more than 1,000 sales per month.
Few shops with a conversion rate above 50.

I thought conversion rate and visibility scores might be good indicators but those had no correlation whatsoever on any other success marker.

What I learned from the pairplots overview.

higher price = less sales makes sense

Views to sales has a somewhat positive linear relationship. So technically if we increase views we'll get more sales though might be opposite with algorithm more sales leads to more views.

auto renews had greater chance for more sales

customization lesser chance for more sales

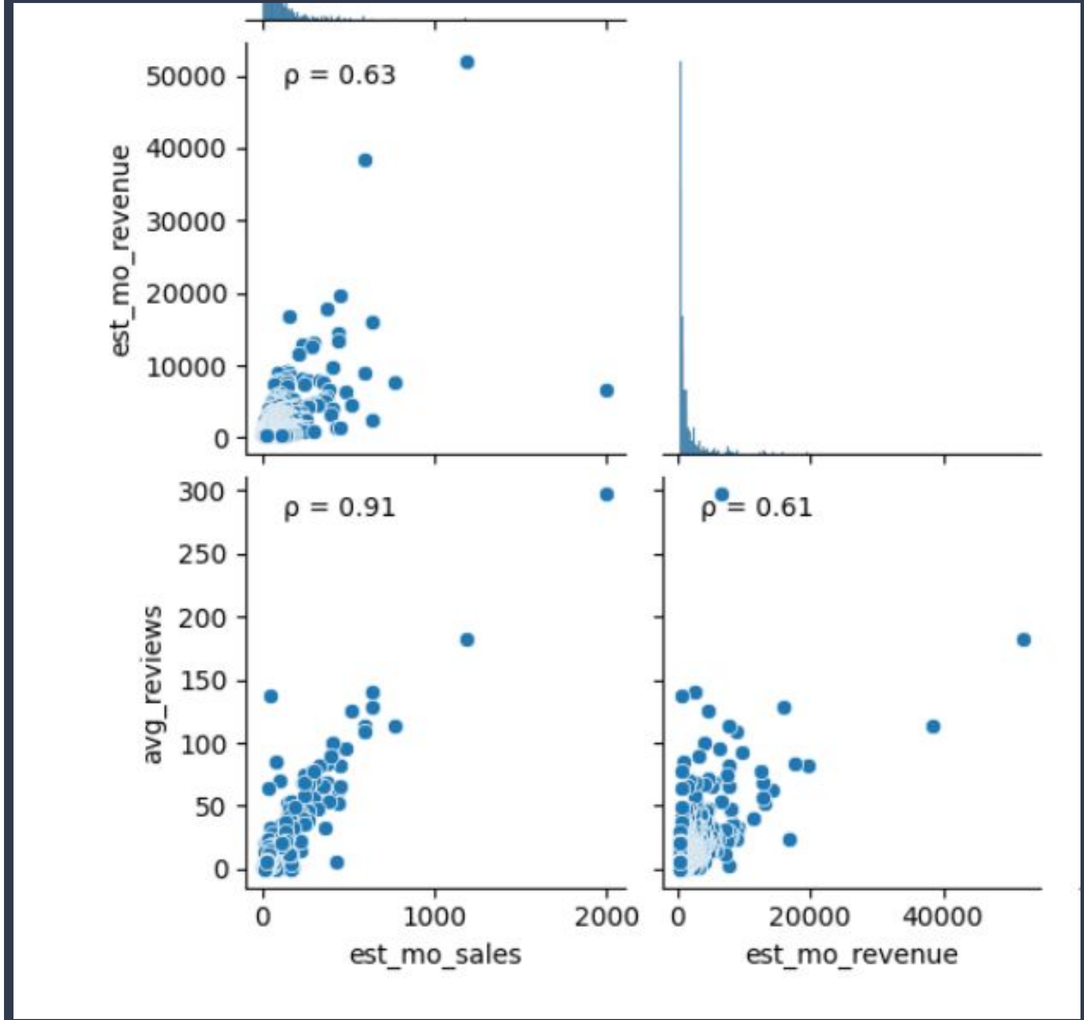
shop age surprising not selling more older shops and surprisingly higher conversion rate does not equal to more sales so not something aim for.

Need to decide which measure to use to see if the product is doing well.

There are few success metrics as total sales avg sales and avg revenue. Looked at all in a pair plot but then decided to look at it on a monthly basis rather than overall because can just be an older shop. I will choose just one metric for success or y variable and drop others.

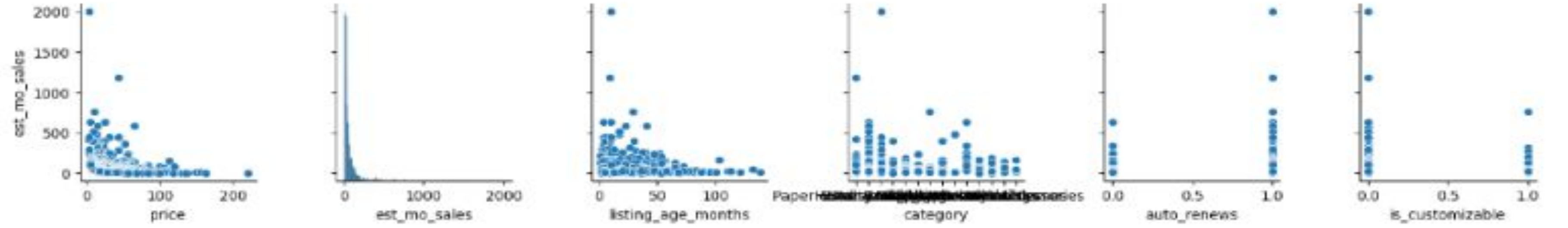
Average reviews and revenue are also good ones but at the end of the day the sale is what matters as higher revenue products might have higher cost and not higher profit.

I thought avg reviews is how many star review is turns out is how many reviews actually got.

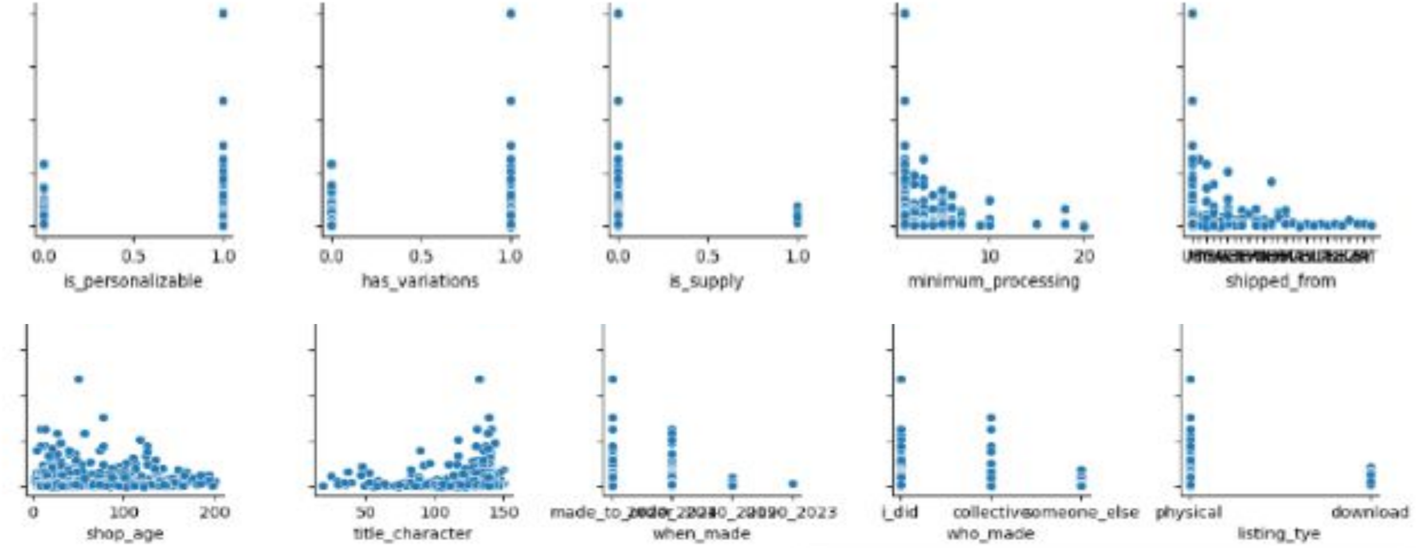


What I learned from the success metrics pairplots.

- More sales and more revenue do not necessarily go hand in hand. Only .63 correlated
- Over a thousand sales from a product per month is an outlier.
- Total sales from product and total shop sales are not at all related.
- More views do lead to more sales and reviews and favorites
- More favorites do lead to more sales
- And reviews are strongest correlation by far. .92 total reviews for total sales. So try what incentives can to get customers to leave reviews. Only question is it more sales that lead to more reviews or more reviews that lead to more sales. Either way pushing reviews is something that pays to explore further.



Comparing the target variable est_mo_sales to all feature variables to see how they affect it.



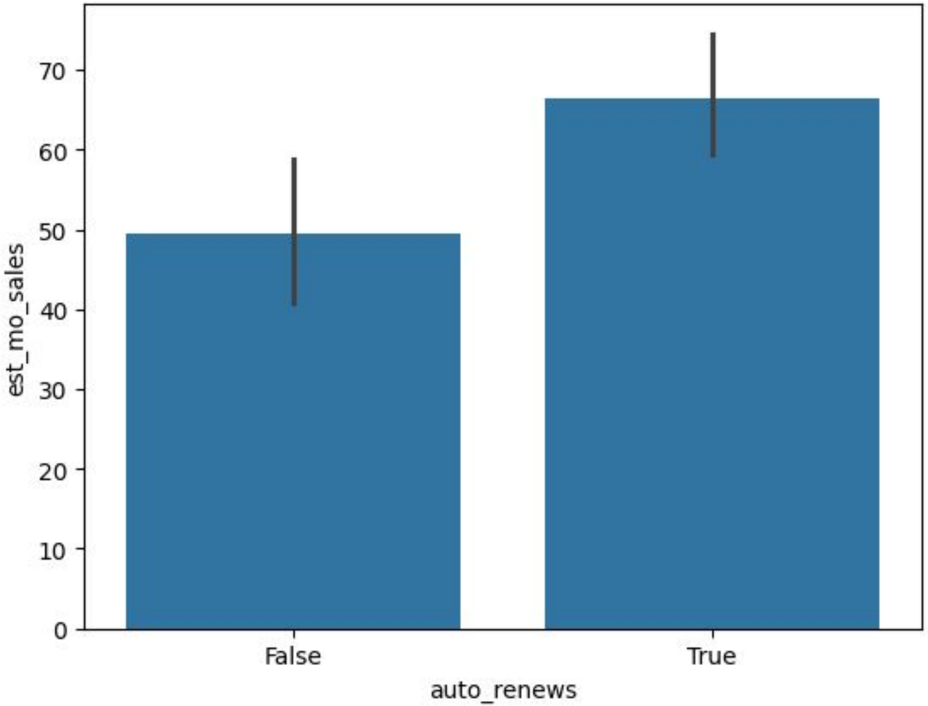
Best selling items are lower priced, more recently listed, they auto renew, are not customizable, are personalizable, do have variations, are not a supply, processing time is shorter, can be anh age shop though usually not very old shop, title characters are more, more recently made and mostly collectively made and is rather a physical product. will do seperate visualizations on some especially category and shipped from bec aren't very visible.

Diagnostic Analytics

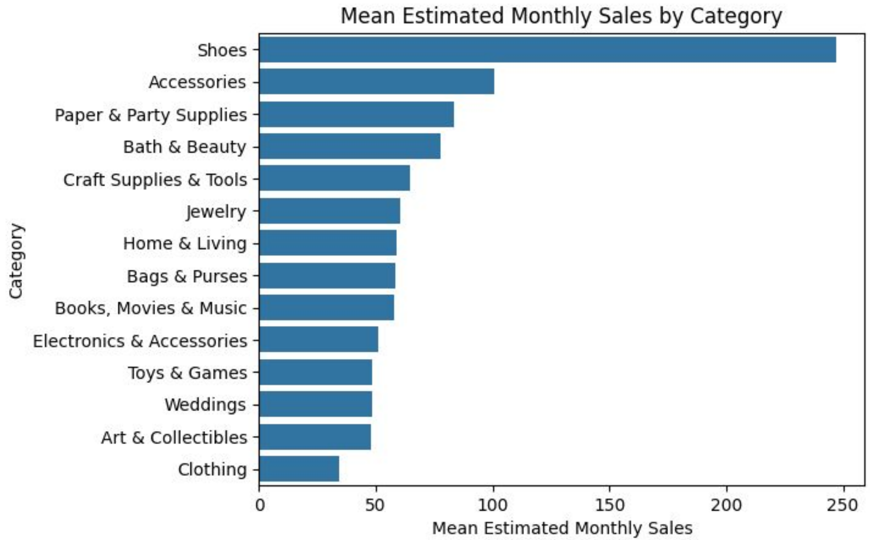
Drilling down to individual questions trying to answer the big question of “**how can we get more sales**?”

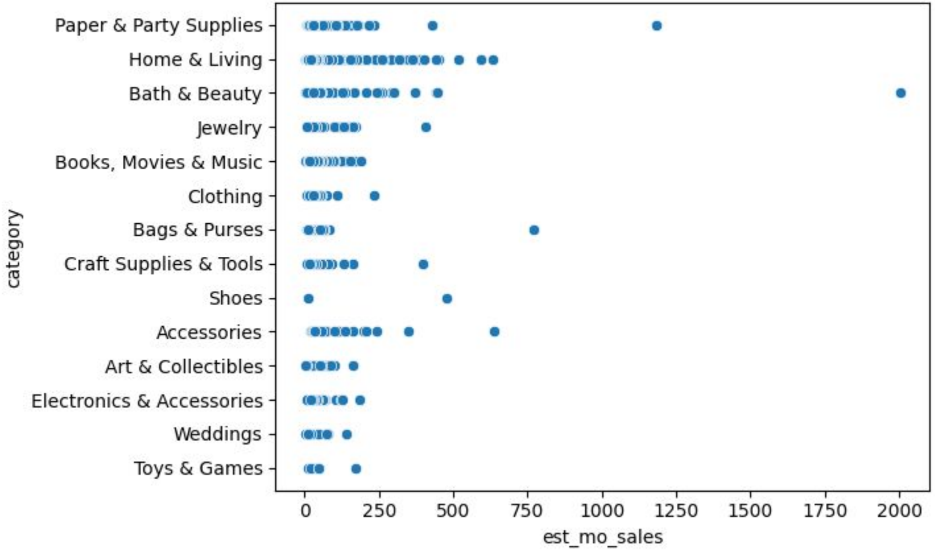
Does auto renewing lead to more sales?

```
#bar plot with x as auto renewing
and y sales
sns.barplot(data=etsy_data,
x='auto_renews',y='est_mo_sales')
plt.show()
#seems like the ones with renewals
have more sales on avg
```



Which category has the most avg sales?



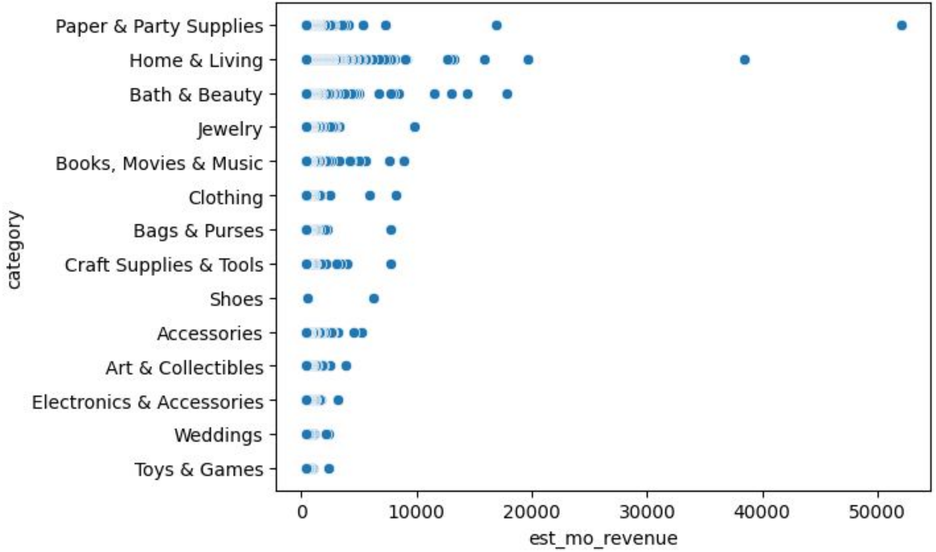


When looking at bar chart seems like pays to branch out to shoes but then on scatterplot see that only two values for shoes that’s why it’s ranking so high. The categories that make more sales on avg without the huge outliers seem to be **home and living**, **bath**, **beauty** and **accessories** categories. Choose products can branch out in one of these categories.

Which category has the highest average revenue?

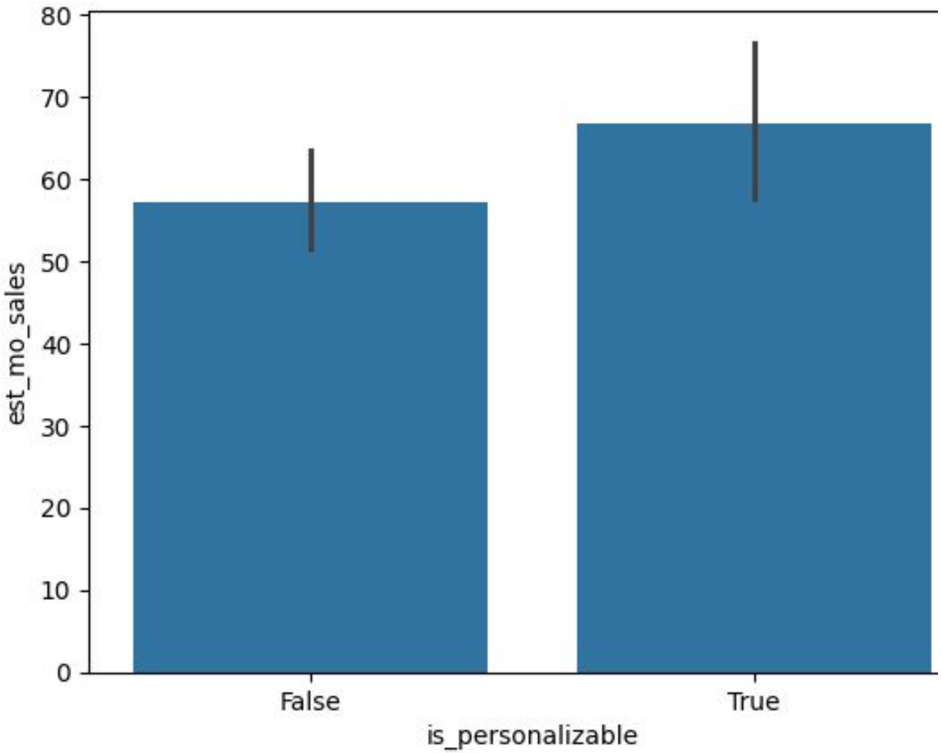
```
#bar plot with y as categories and y
rev
sns.scatterplot(data=etsy_data,
y='category',x='est_mo_revenue')
#plt.xticks(rotation=45)
plt.show()
```

When looking at revenue instead of sales the results are pretty similar so in future will just look at sales over revenue.



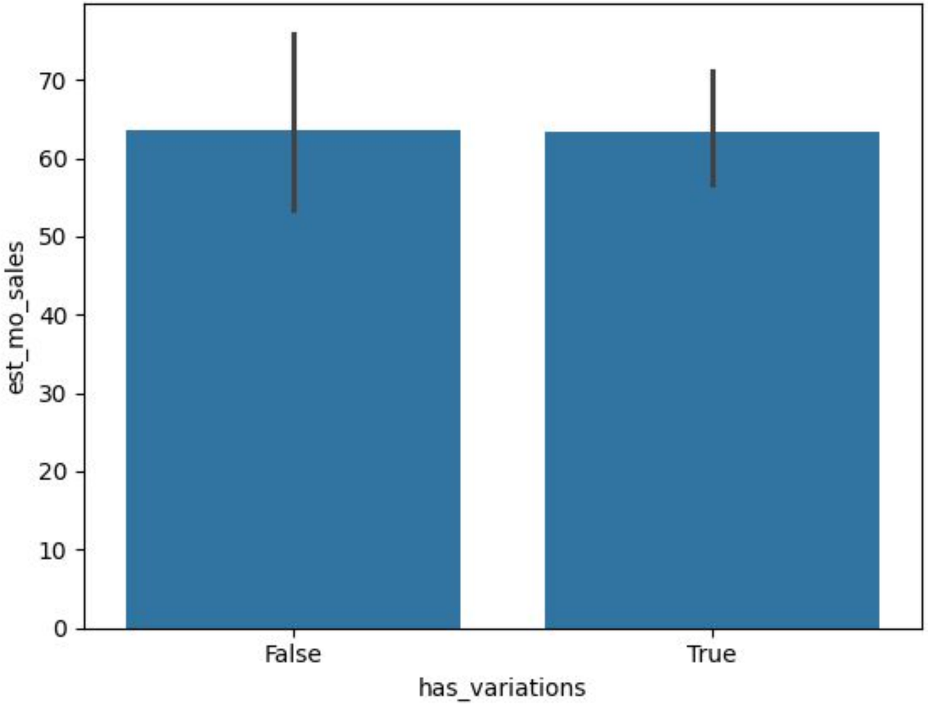
Do personalizable listings lead to more sales?

```
bar plot with x as personalizable and y
sales
sns.barplot(data=etsy_data,
y='est_mo_sales',x='is_personalizable')
plt.show()
#seems like it does lead to a bit more
sales not by much and the confidence
interval ia also bigger so don't think
it pays to focus on personalizable
listings so much as it does take a lot
more time and harder to automate while
not leading to a significant enough
increase in sales on average.
```

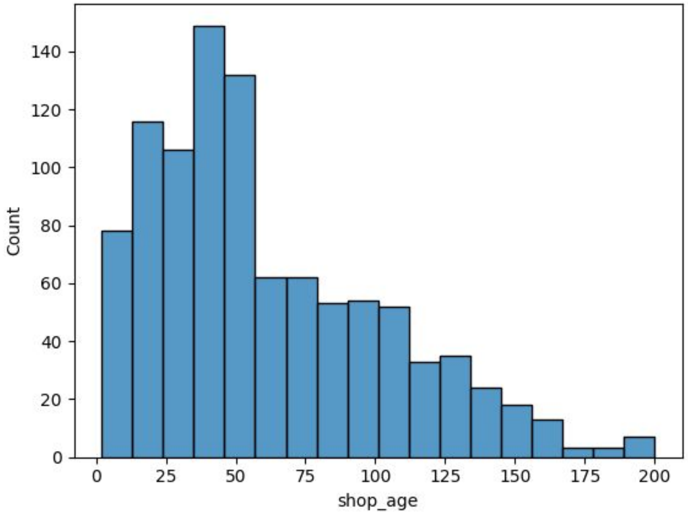


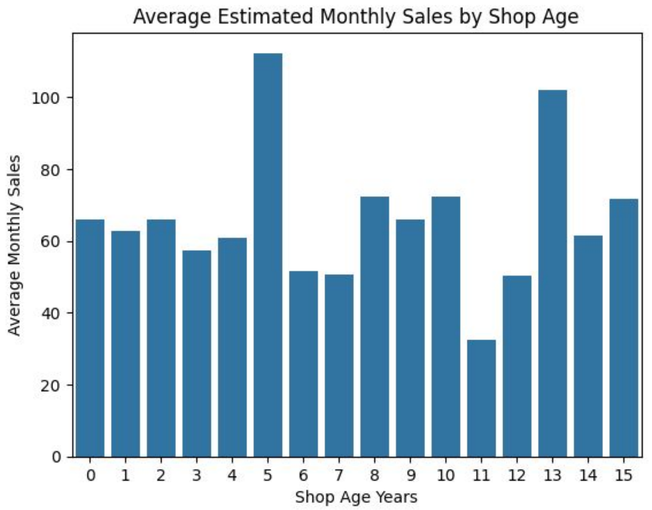
Do listings with variations have more sales?

```
sns.barplot(data=etsy_data,x=
'has_variations', y='est_mo_sales')
plt.show()
#there seems to be no difference on
avg if a product has variations or
not. Though products with no
variations do have a bigger
confidence interval.
```



How old are most shops and do older shops have more sales?

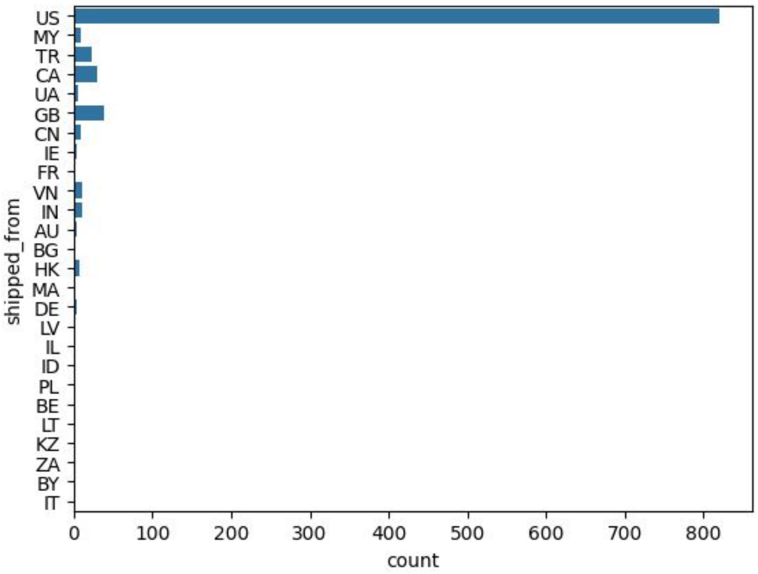


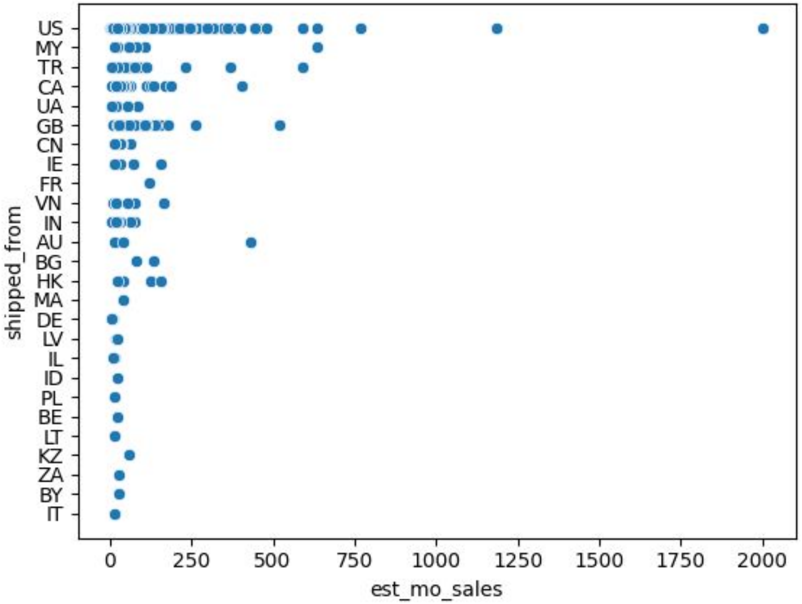


There seems to be many more newer shops with majority between 40 and 50 months old that is around 4 years old. There was a huge spike of new shops at covid time. It has dropped since then but still higher than precovid new shops per year.

The shops that opened right before covid seem to be doing best now, with ones opened 13 years ago also. Otherwise shops opened in the last few years seem to be doing about the same on avg and even better than some really old shops. So opening a new second shop now shouldn't be an issue.

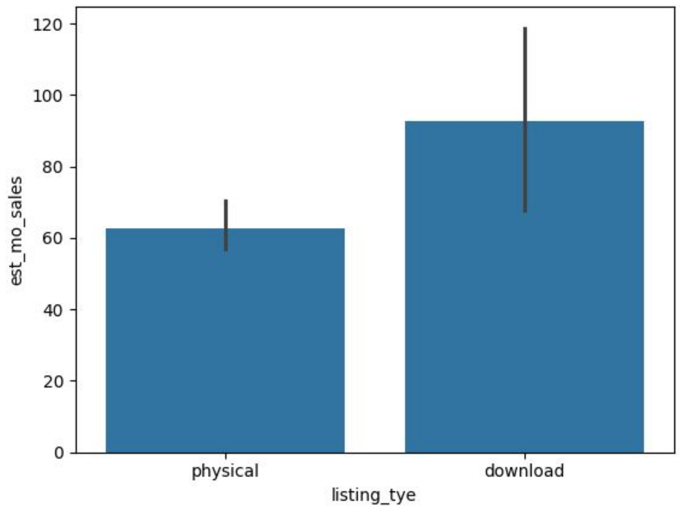
Which country are most shipped from?
Does any shipping country have higher average sales?

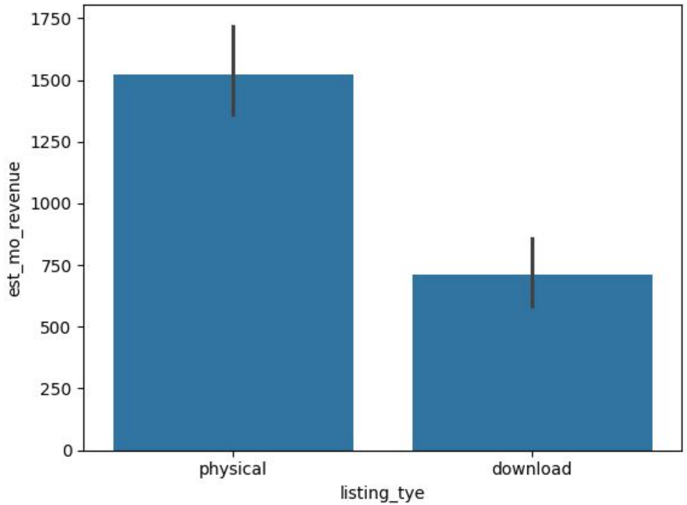




Most are shipped from US and ones with higher sales are also US. So won't pursue shipping providers outside of US for now. Although gb- Great Britain and Canada seem like ranking next outside of one off outliers if ever want to expand.

Do digital listings have more sales?

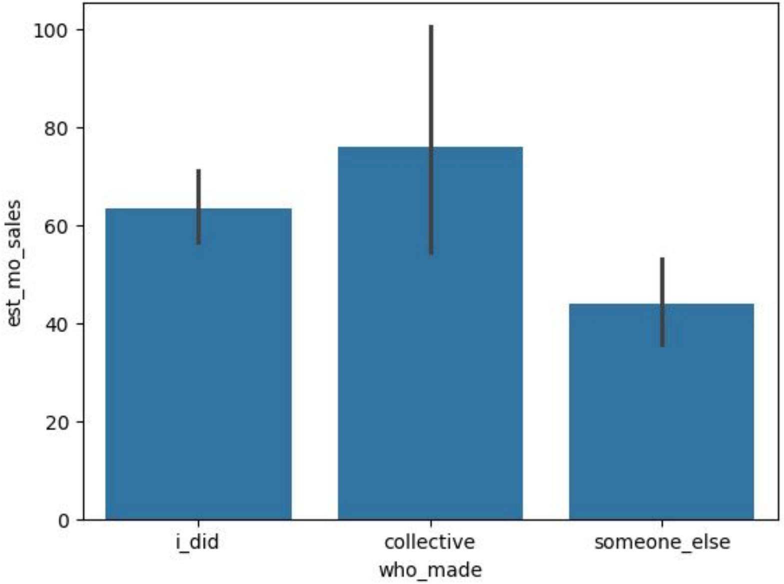


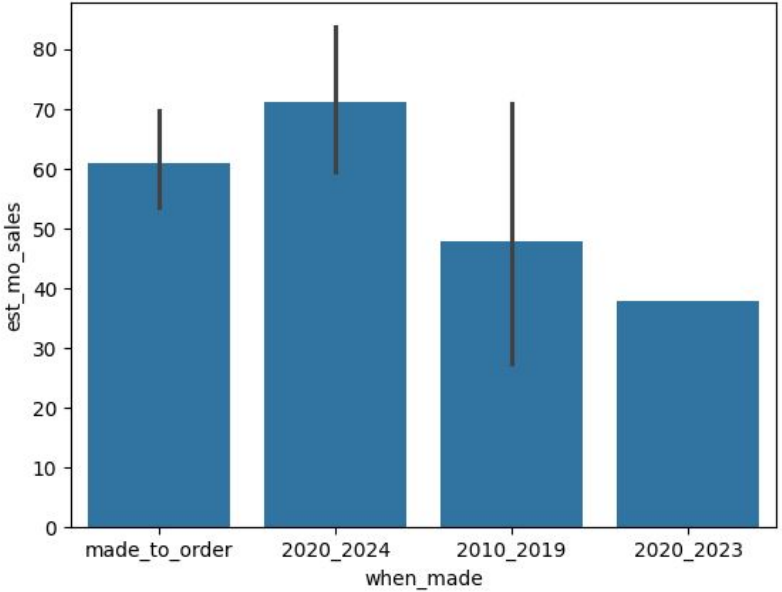


Totally unexpected but digital listing on avg have more sales. Will check revenue as well if also more since digital listings are usually cheaper. maybe pays to expand more to downloads.

So although digital products make many more sales on average the average revenue is much less than for physical products.

Does who made it, or when it was made impact sales?





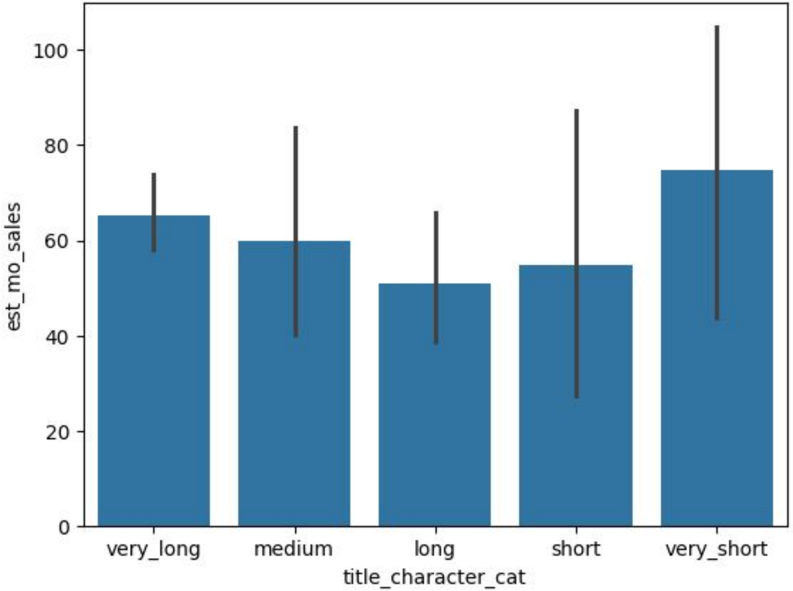
Collectively made has higher avg sales. So pays to stay in the pod model = collectively made as producing on own does not seem to lead to more sales.

Seems like 2024 first with made to orders second. Although it doesn't pay for us to manufacture in advance it might be viable once it has high enough sales or might pursue one item to make in advance in bulk as it can then sell at lower cost might lead to more sales.

Do more title characters (using every character in title allowed) lead to more sales?

Grouped the title character length in 5.

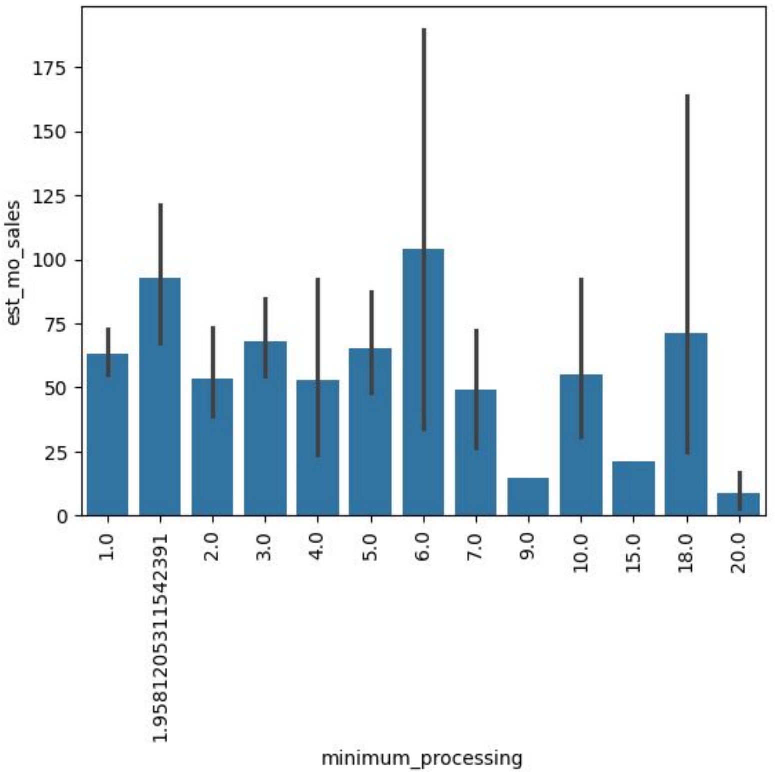
```
#see the range
title_len = etsy_data['title_character'].unique()
print(sorted(title_len))
#from 19-150
# Define the conditions and corresponding categories
conditions = [
    (etsy_data['title_character'] <= 45),
    (etsy_data['title_character'] > 45) & (etsy_data['title_character'] <= 71),
    (etsy_data['title_character'] > 71) & (etsy_data['title_character'] <= 97),
    (etsy_data['title_character'] > 97) & (etsy_data['title_character'] <= 123),
    (etsy_data['title_character'] > 123) & (etsy_data['title_character'] <= 150)
]
categories = ['very_short', 'short', 'medium', 'long', 'very_long']

# Create the new column based on conditions
etsy_data['title_character_cat'] = np.select(conditions, categories,
default='extra_long')
# Display the DataFrame head to verify the new column
print(etsy_data.head())
```



Totally contrary to popular belief seems like using every title character allowed does not necessarily lead to more sales on avg. In fact very short titles has many more sales on avg although the confidence interval for that is very large. Should do a-b split testing same listing one with title under 45 characters and other that uses all possible character spaces and see how they do regarding views.

Do lower processing time lead to more sales?

Longer processing time (>6 days) does seem to lead to less avg sales. The second column is all ones with missing values. Seems to not be a very realistic replacement. One with 6 days has a very big error bar so not very accurate. Will change minimum shipping to 1 day bec theoretically it's possible and see if it leads to uptick in sales.



Predictive Analytics

Try different models to see which can best predict factors that result in higher sales.

Filter to get only rows where category is not a supply since supplies is not something we can do

```
etsy_data_testing = etsy_data_testing[etsy_data_testing['is_supply']==False]
#validate that craft supplies are not there anymore
print(etsy_data_testing['is_supply'].unique())
print(etsy_data_testing.shape)
#can drop that column now since all are now not supply
etsy_data_testing = etsy_data_testing.drop('is_supply',axis = 1)
#validate that one column less now
print(etsy_data_testing.shape)
```

Get Dummy variables for categorical columns

```
from sklearn.preprocessing import OneHotEncoder
dummies = OneHotEncoder()
dummy_array = dummies.fit_transform(etsy_data_testing[['category', 'who_made', 'listing_tye']]).toarray()
#add title for each column
prefixes = ['category', 'who_made', 'listing_tye']
dummy_labels = dummies.categories_
labels = np.array([f'{prefix}_{label}' for prefix, sublist in zip(prefixes, dummy_labels) for label in sublist])
labels = np.array(labels)
dummy = pd.DataFrame(dummy_array, columns = labels)
#recombine dummy variables with continuous ones
etsy_data_testing = pd.concat([etsy_data_testing[['price', 'est_mo_sales',
       'listing_age_months','auto_renews', 'is_customizable', 'is_personalizable',
        'has_variations', 'minimum_processing', 'shop_age',
       'title_character']],dummy],axis=1)
print(etsy_data_testing.head())
```

Convert the objects/ booleans to numeric

```
#convert the objects/ booleans to
numeric
etsy_data_testing[['auto_renews',
        'is_customizable',
'is_personalizable', 'has_variations']]
= etsy_data_testing[['auto_renews',
        'is_customizable',
'is_personalizable',
'has_variations']].astype(int)

print(etsy_data_testing.dtypes)
```

```
price                                    float64
est_mo_sales                             float64
listing_age_months                       float64
auto_renews                                int64
is_customizable                            int64
is_personalizable                          int64
has_variations                             int64
minimum_processing                       float64
shop_age                                 float64
title_character                          float64
category_Accessories                     float64
category_Art & Collectibles              float64
category_Bags & Purses                   float64
category_Bath & Beauty                   float64
category_Books, Movies & Music           float64
category_Clothing                        float64
category_Craft Supplies & Tools          float64
category_Electronics & Accessories       float64
category_Home & Living                   float64
category_Jewelry                         float64
category_Paper & Party Supplies          float64
category_Shoes                           float64
category_Toys & Games                    float64
category_Weddings                        float64
category_nan                             float64
who_made_collective                      float64
who_made_i_did                           float64
who_made_someone_else                    float64
listing_tye_download                     float64
listing_tye_physical                     float64
```

Split model for testing and training.

```
from sklearn.model_selection import
train_test_split
# Split data into features (X) and target
variable (y)
X =
etsy_data_testing.drop(columns=['est_mo_sales']
)
y = etsy_data_testing['est_mo_sales']
# Split id's into training and testing sets for
when split data
X_train, X_test, y_train, y_test =
train_test_split(X,y, test_size=0.25,
random_state=42)
```

Import different models to try.

```
#linear regression
from sklearn.linear_model import LinearRegression
#gradient boosting
from sklearn.ensemble import
GradientBoostingRegressor
# decision tree
from sklearn.tree import DecisionTreeRegressor
#ridge regression
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error,
r2_score
```

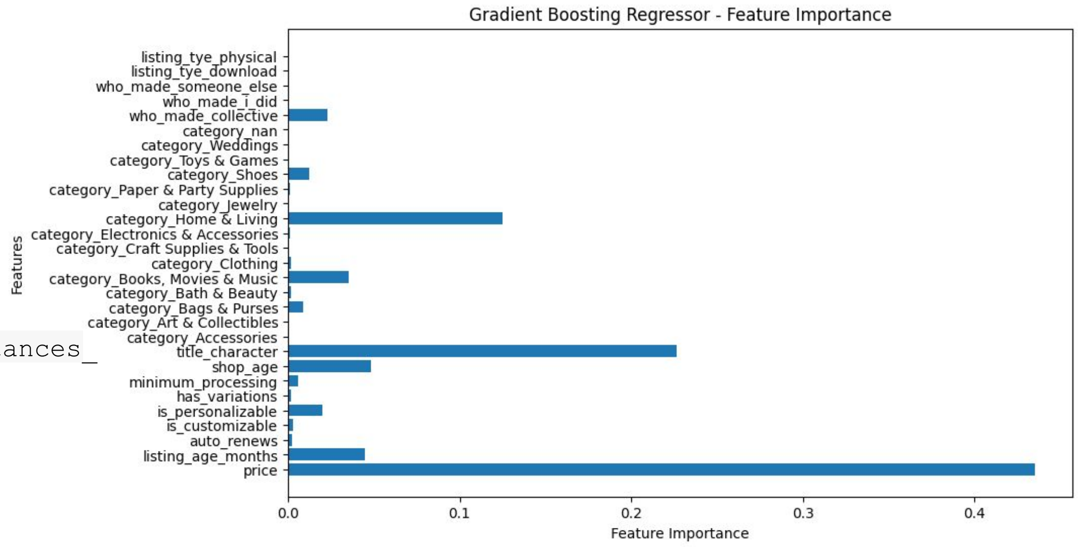
Run Models

```python
#list of models
models=[LinearRegression(),GradientBoostingRegressor(),DecisionTreeRegressor(), Ridge()]
#empty list to store avg precision
mse_scores = []
r_squared_scores = []

#run models
for model in models:

    #fit the model
    model.fit(X_train,y_train)
    y_pred = model.predict(X_test)
    # Compute Mean Squared Error (MSE)
    mse = mean_squared_error(y_test, y_pred)
    mse_scores.append(mse)

    # Compute R-squared (R2) score
    r_squared = r2_score(y_test, y_pred)
    r_squared_scores.append(r_squared)

# Create a dataframe of the average precision of each model
df_scores = pd.DataFrame({Model': models, 'MSE': mse_scores, 'R-squared' : r_squared_scores})

# Print the data frame
print(df_scores)
```

	Model	MSE	R-squared
0	LinearRegression()	8362.608644	0.025642
1	([DecisionTreeRegressor(criterion='friedman_ms...	8846.795687	-0.030772
2	DecisionTreeRegressor()	23928.367816	-1.787980
3	Ridge()	8060.044366	0.060895

Gradient Boosting Regressor Optimization. Get the important feature.

```python
# Instantiate the model
gb_regressor = GradientBoostingRegressor()

# Fit the model to your data
gb_regressor.fit(X_train, y_train)

# Get feature importances
feature_importance = gb_regressor.feature_importances_

# Plot feature importances
plt.figure(figsize=(10, 6))
plt.barh(X_train.columns, feature_importance)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Gradient Boosting Regressor - Feature Importance')
plt.show()
```



Price seems to be the most important indicator with title character next. Age also seems to predict sales a bit.

Reducing Features

```
# Get indices of significant features (assuming you want to
keep features with non-zero importance)
significant_indices = feature_importance > 0

# Filter the training data to include only significant
features
X_train_filtered = X_train.iloc[:, significant_indices]
X_test_filtered = X_test.iloc[:, significant_indices]
# Retrain the gradient boosting model with filtered features
gb_model_filtered = GradientBoostingRegressor()
gb_model_filtered.fit(X_train_filtered, y_train)

# Make predictions
y_pred_gb_model_filtered =
gb_model_filtered.predict(X_test_filtered)
r_squared_gb_model_filtered = r2_score(y_test,
y_pred_gb_model_filtered)
print(r_squared_gb_model_filtered)
```

Reducing features improved the gradient boosting model to .15 though When ran it next day actually performed much worse at only .06

Bagging

```
#will try to do a bagging regressor to see if get better
results
from sklearn.ensemble import BaggingRegressor

# Instantiate a base gradient boosting regressor model
base_model = GradientBoostingRegressor()

# Instantiate a BaggingRegressor with the base model
bagging_model = BaggingRegressor(base_model,
n_estimators=10, random_state=42)

# Train the bagging model
bagging_model.fit(X_train, y_train)

# Make predictions
y_pred_bagging = bagging_model.predict(X_test)
r_squared_bagging = r2_score(y_test, y_pred_bagging)
print(r_squared_bagging)
```

Bagging improved the r squared to .18

Model is better but still not good enough rsquared is still under .2 will try to do linear model on own.

```
#try fitting linear model to see which
factors matter
import statsmodels.api as sm
X_train = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train).fit()
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           est_mo_sales   R-squared:                       0.133
Model:                            OLS   Adj. R-squared:                  0.088
Method:                 Least Squares   F-statistic:                     2.922
Date:                Sun, 07 Apr 2024   Prob (F-statistic):           2.99e-06
Time:                        02:32:15   Log-Likelihood:                -3206.0
No. Observations:                 522   AIC:                             6466.
Df Residuals:                     495   BIC:                             6581.
Df Model:                          26
Covariance Type:            nonrobust
=====================================================================================================
                                        coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------------
const                                44.5052     24.128      1.845      0.066      -2.900      91.911
price                                -1.1371      0.226     -5.037      0.000      -1.581      -0.694
listing_age_months                   -0.2465      0.326     -0.756      0.450      -0.887       0.394
auto_renews                          27.2622     13.288      2.052      0.041       1.155      53.369
is_customizable                     -30.5278     16.397     -1.862      0.063     -62.744       1.688
is_personalizable                    25.7443     11.420      2.254      0.025       3.307      48.182
has_variations                       -5.9752     14.906     -0.401      0.689     -35.261      23.311
minimum_processing                   -1.0076      2.159     -0.467      0.641      -5.250       3.235
shop_age                              0.2157      0.148      1.460      0.145      -0.075       0.506
title_character                      -0.0277      0.261     -0.106      0.915      -0.540       0.484
category_Accessories                 -2.7824     25.979     -0.107      0.915     -53.825      48.261
category_Art & Collectibles          -8.6339     33.362     -0.259      0.796     -74.183      56.915
category_Bags & Purses               -8.8191     26.766     -0.329      0.742     -61.409      43.771
category_Bath & Beauty                4.2382     32.592      0.130      0.897     -59.798      68.274
category_Books, Movies & Music       96.9387     28.381      3.416      0.001      41.177     152.700
category_Clothing                   -21.2429     26.608     -0.798      0.425     -73.521      31.035
category_Craft Supplies & Tools     -26.2377     33.300     -0.788      0.431     -91.665      39.190
category_Electronics & Accessories  -37.0528     28.717     -1.290      0.198     -93.475      19.369
category_Home & Living               -7.7946     14.322     -0.544      0.587     -35.935      20.345
category_Jewelry                    -31.5782     24.631     -1.282      0.200     -79.972      16.816
category_Paper & Party Supplies     -25.7822     21.984     -1.173      0.241     -68.975      17.411
category_Shoes                      217.5941    109.521      1.987      0.047       2.410     432.778
category_Toys & Games               -47.5954     42.902     -1.109      0.268    -131.888      36.697
category_Weddings                   -38.6523     42.879     -0.901      0.368    -122.899      45.594
category_nan                        -18.0944    109.861     -0.165      0.869    -233.945     197.756
who_made_collective                  50.8960     15.095      3.372      0.001      21.238      80.554
who_made_i_did                       14.3140     10.829      1.322      0.187      -6.962      35.590
who_made_someone_else               -20.7049     16.736     -1.237      0.217     -53.586      12.177
listing_tye_download                 18.8891     25.737      0.734      0.463     -31.678      69.456
listing_tye_physical                 25.6161     17.550      1.460      0.145      -8.866      60.098
==============================================================================
Omnibus:                      812.943   Durbin-Watson:                   2.027
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           316545.159
Skew:                           8.615   Prob(JB):                         0.00
Kurtosis:                     122.402   Cond. No.                     4.00e+16
==============================================================================
```

Optimized Regression Results

```python
# Filter the dataset to include only statistically significant features
significant_features = model.pvalues[model.pvalues < 0.05].index
X_train_filtered = X_train[significant_features]

# Re-run the linear regression model with filtered features
model_filtered = sm.OLS(y_train, X_train_filtered).fit()
print(model_filtered.summary())
```

Managed to improve the model to .32 r squared after leaving only the few significant categories.

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:            est_mo_sales   R-squared (uncentered):                   0.323
Model:                             OLS   Adj. R-squared (uncentered):              0.315
Method:                  Least Squares   F-statistic:                              40.96
Date:                 Sun, 07 Apr 2024   Prob (F-statistic):                    8.12e-41
Time:                         02:32:18   Log-Likelihood:                         -3224.3
No. Observations:                  522   AIC:                                      6461.
Df Residuals:                      516   BIC:                                      6486.
Df Model:                            6
Covariance Type:             nonrobust
=================================================================================================
                                    coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------------
price                            -0.7869      0.208     -3.788      0.000      -1.195      -0.379
auto_renews                      68.4015      9.167      7.462      0.000      50.392      86.411
is_personalizable                38.3758     10.181      3.769      0.000      18.375      58.377
category_Books, Movies & Music  100.6391     26.819      3.752      0.000      47.951     153.327
category_Shoes                  238.1075    117.423      2.028      0.043       7.422     468.793
who_made_collective              44.8749     16.136      2.781      0.006      13.174      76.575
==============================================================================
Omnibus:                      782.554   Durbin-Watson:                   2.044
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           265248.099
Skew:                           8.026   Prob(JB):                         0.00
Kurtosis:                     112.260   Cond. No.                         862.
==============================================================================
```

Price, autorenews, is personalizable and **who made are** the most important features.

- Higher price lowers the chance of selling one unit more slightly by .79
- Turning auto renewal on increases chance of selling more by 68
- Being personalizable items increase chance of selling more by 38
- Being collectively made also increases chance of selling by around 45

Model is still far from perfect even though managed to improve it. When have more time will definitely need to iterate again remove the outliers and maybe narrow down to one category.

Prescriptive Analytics

recommend actions:

1. Find products to sell in home and living, bath and beauty, or accessories categories.
2. Try different incentives to get customers to leave reviews.
3. Turn auto renewal on.
4. Add personalizable products.
5. Do lower price or at least a loss leader- one unpopular variation selling for less.
6. Do a-b split testing same listing one with title under 45 characters and other that uses all possible character spaces and see how they do regarding views.
7. Change minimum shipping to 1 day.