ITP 449 Machine Learning Final Project:
Predicting Camera Prices

Introduction

For this project, I will be analyzing a dataset of cameras comprised of technical specifications and prices. I want my model to use the technical specifications as inputs to predict the target variable price. I will be exploring both classification and regression models to find the most accurate predictor.

Here is a snapshot of the dataset:

	Model	Release date	Max resolution	Low resolution	Effective pixels	Zoom wide (W)	Zoom tele (T)	Normal focus range	Macro focus range	Storage included	Weight (inc. batteries)	Dimensions	Price
0	Agfa ePhoto 1280	1997	1024.0	640.0	0.0	38.0	114.0	70.0	40.0	4.0	420.0	95.0	179.0
1	Agfa ePhoto 1680	1998	1280.0	640.0	1.0	38.0	114.0	50.0	0.0	4.0	420.0	158.0	179.0
2	Agfa ePhoto CL18	2000	640.0	0.0	0.0	45.0	45.0	0.0	0.0	2.0	0.0	0.0	179.0
3	Agfa ePhoto CL30	1999	1152.0	640.0	0.0	35.0	35.0	0.0	0.0	4.0	0.0	0.0	269.0
4	Agfa ePhoto CL30 Clik!	1999	1152.0	640.0	0.0	43.0	43.0	50.0	0.0	40.0	300.0	128.0	1299.0

Data Cleaning

For this dataset, I was only interested the technical specifications as predictors, so I removed the model name and the release date of the camera. It is important to note that this already leaves room to skew the model due to the price of a camera also being attached to the release date and the brand, considering the value of vintage and the prestige of a name. I am mostly curious to see if the model will be accurate without this information. I think that vintage value and brand name as more arbitrary and random of predictors and wouldn't be particularly valuable in this context.

I also had to remove rows with null values as well as clear white spaces for column headers.

At this stage, I also created a separate data frame that converted prices into categories A-H, each category responding to a price range. I did this for the purposes of observing classification models as well as regression. The categories are arbitrary and directly influence the model, but the ranges that I chose would be highly applicable for the purpose of marketing cameras.

I then split the data into training and testing. I utilized dummy classifiers to observe a reasonable baseline model for further comparison. I then tuned the train_test_split function to contain parameters that maximized this baseline accuracy. It was found to be around 32% accurate.

Baseline Model Testing

For regression models, I am using MSE as a comparison metric. For classification models, I will be using sklearn.metrics accuracy

1. **Linear Regression**

Predictions on the train set
MSE: 418817.11354437156
Coefficient of R^2: 0.3123049928185816

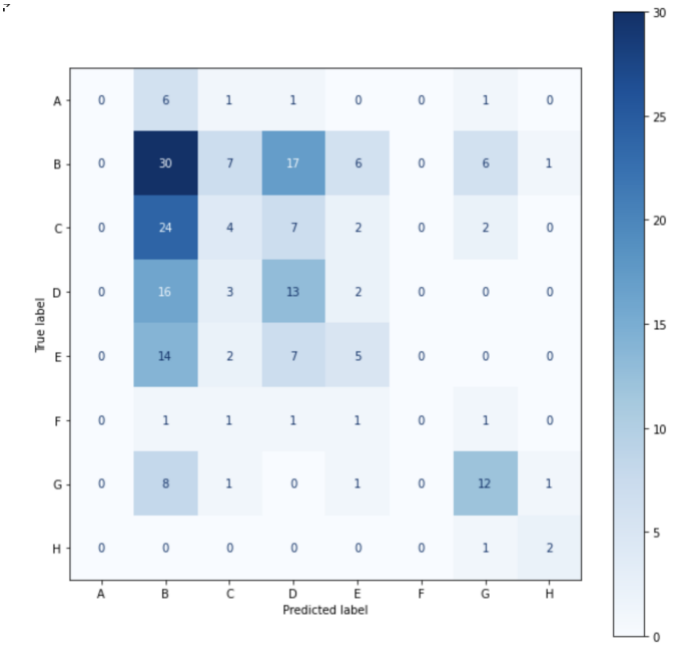
Predictions on the test set
MSE: 356128.28431307455
Coefficient of R^2: 0.22058047846385465

2. **Logistic Regression**

Confusion Matrix:

```
[[ 0  6  1  1  0  0  1  0]
 [ 0 30  7 17  6  0  6  1]
 [ 0 24  4  7  2  0  2  0]
 [ 0 16  3 13  2  0  0  0]
 [ 0 14  2  7  5  0  0  0]
 [ 0  1  1  1  1  0  1  0]
 [ 0  8  1  0  1  0 12  1]
 [ 0  0  0  0  0  0  1  2]]
```

Accuracy = 0.3173076923076923



3. Decision Tree Classification

Confusion Matrix:
[[6 0 0 1 1 0 1 0]
[0 42 7 7 6 1 3 1]
[0 5 21 6 4 1 1 1]
[1 4 2 20 2 0 5 0]
[1 4 2 3 17 0 1 0]
[0 0 0 3 0 0 2 0]
[0 0 0 5 2 1 15 0]
[0 0 0 0 0 0 1 2]]
Accuracy = 0.5913461538461539

4. Bagging

Using Decision Tree Classifier as the base model as it is the most promising.

BaggingClassifier(DecisionTreeClassifier(max_depth=20, criterion = 'entropy', random_state=42), random_state=2020, n_estimators=100)

Accuracy = 0.6586538461538461

The parameters were set with trial and error on maximizing the accuracy score.

5. **Random Forest**

```
RandomForestClassifier(n_estimators=300, max_features=8, random_state=42)
```

Accuracy score 0.6682692307692307

```
Important Feature
Weight_(inc._batteries)    0.167906
Dimensions                 0.150372
Zoom_tele_(T)              0.108179
Storage_included           0.103517
Normal_focus_range         0.097833
Macro_focus_range          0.094986
Low_resolution             0.094254
Zoom_wide_(W)              0.068729
Max_resolution             0.068391
Effective_pixels           0.045832
dtype: float64
```

6. **ADA Boosting**

```
VotingClassifier(estimators = [('RF', forest), ('SVM', SVM), ('REG', regression), ('DT', dt)],
voting='soft')
```

Accuracy score 0.6009615384615384

7. **XGB Regression**

```
xg_reg = xgb.XGBRegressor(objective ='reg:linear', colsample_bytree = 0.2, learning_rate = 0.1,
            max_depth = 5, alpha = 10, n_estimators = 100)
```

MSE: 168503.483915

8. **XGB Classification**

```
xg_reg = xgb.XGBClassifier(colsample_bytree = 0.2, learning_rate = 0.2,
            max_depth = 40, alpha = 10, n_estimators = 100)
```

Accuracy: 0.6778846153846154

After messing with the parameters of each model to see if there were any intuitive improvements to be made to any of these models, it was clear the XGBoost was the most promising of all sets, against both the regression and classification performance benchmarks. This is the model I moved forward with in further cross validation and regularization.

Chosen Model Exploration/Initial Tuning

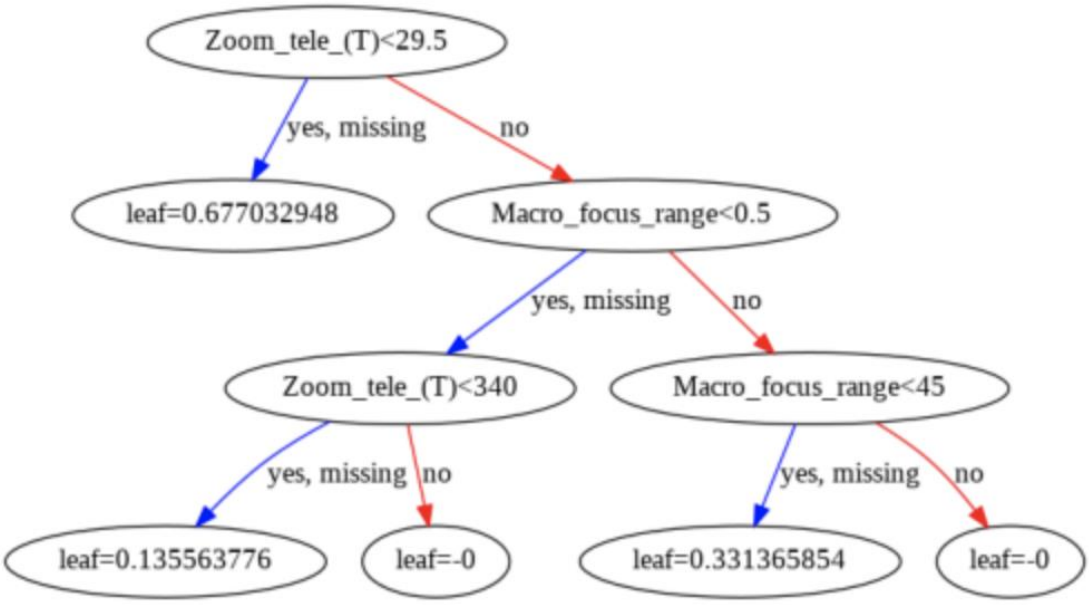
After deciding that XGBoost was the most promising model and fine tuning the parameters through trial and error, I wanted to do some initial cross validation to observe the success of a low effort attempt at the model. I used a label encoder to convert the categorical definition of price (A-H) to a numerical value (0-7). This was the initial CV with that data set.

```
params = {'colsample_bytree' : 0.2, 'learning_rate' : 0.2,
          'max_depth' : 40, 'alpha' : 10, 'n_estimators' : 100}

cv_results = xgb.cv(dtrain=data_dmatrix, params=params, nfold=3,
                    num_boost_round=999,early_stopping_rounds=10, as_pandas=True, seed=123,
metrics={'mae'})

cv_results['test-mae-mean'].min()
```

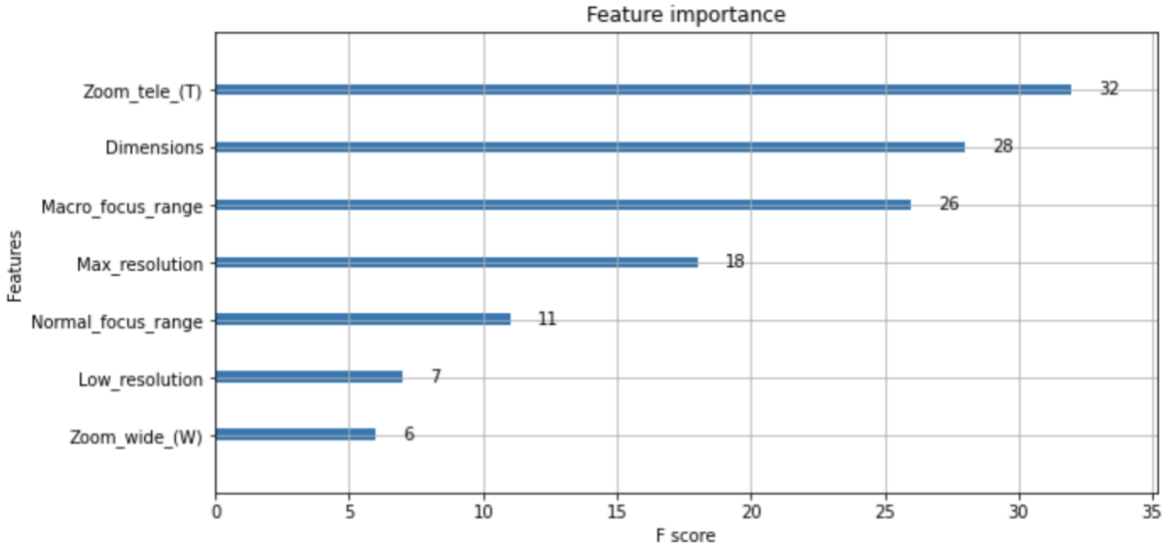
Best MAE: 1.1138096666666666

The best iteration of the model still, on average, was over a full category off in its prediction.

Here is the first tree representing this model:



Initial Feature Importance:



This was mostly for the purpose of exploring the initial model further. Nothing is clearly being revealed, but it has set up the methods for further regularization through quantitative methods.

Hyperparameter Tuning XGB

To properly regularize the model through cross-validation, I needed to tune the parameters to find the lowest MAE iteration within a range of given parameters. I utilized for loops to run associated combinations of parameters for optimizing MAE. Prior to running these loops, I decided it didn't make sense to change a regression dataset into a classification set and then back, so I used the train_test_split on the original data set with the target variable price in terms of dollars. This means that MAE reflects dollar amount of the average. The parameter combinations I used were (max depth + min child weight), (subsample + col sample), and ETA. I trained the regression model initially to find the benchmark numbers to compare the changes of optimization.

Without XGB, Baseline MAE is 402.86.

With default XGB parameters, Best MAE is 223.834 with 5 rounds and min test MAE is 254.35144959999997.

These will be the reference for improvement.

For Max Depth and Min Child Weight respectively: best params: 8, 1, MAE: 236.7938048

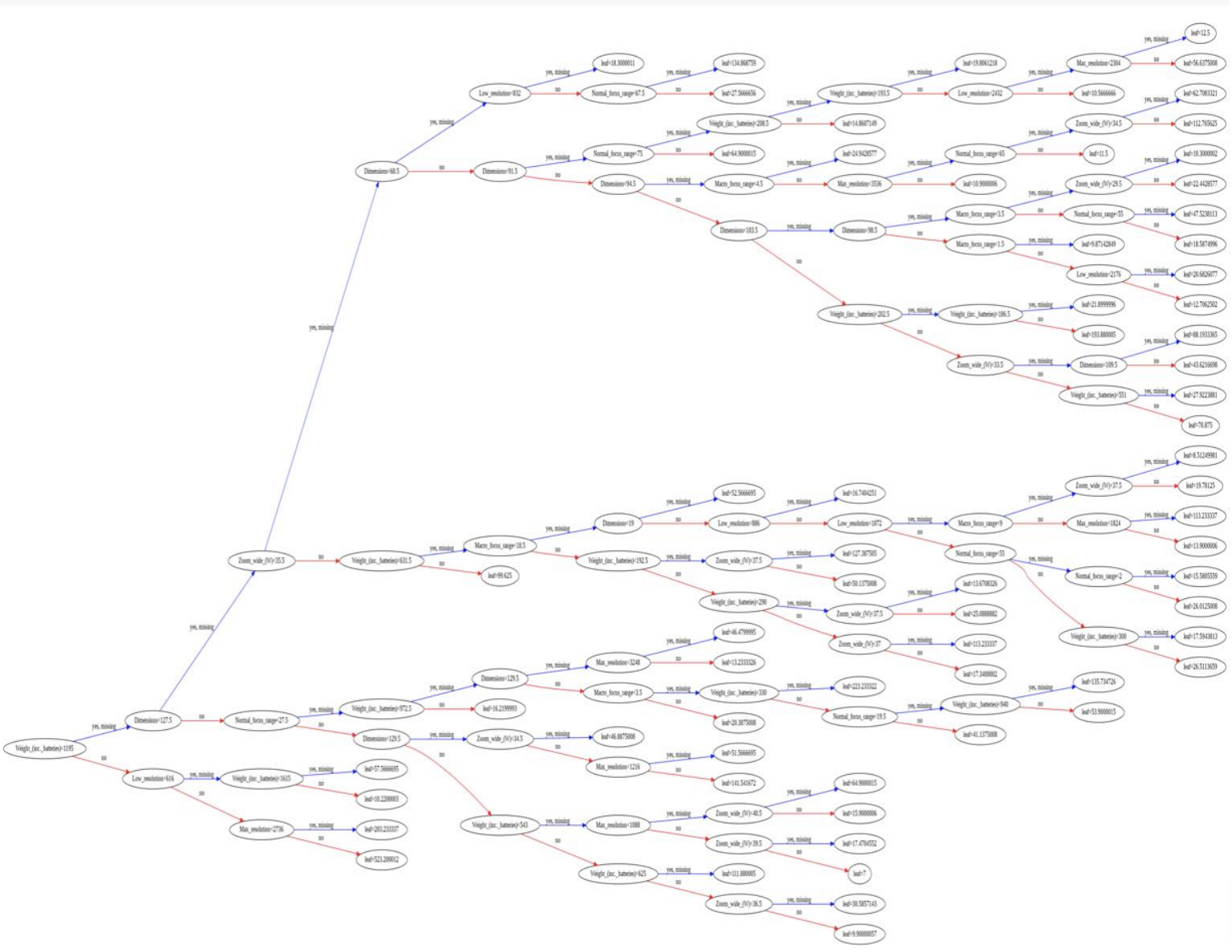
For Subsample and Col sample respectively: best params: 0.9, 1.0, MAE: 242.4313874

For ETA: best params: 0.1, MAE: 232.89493739999998

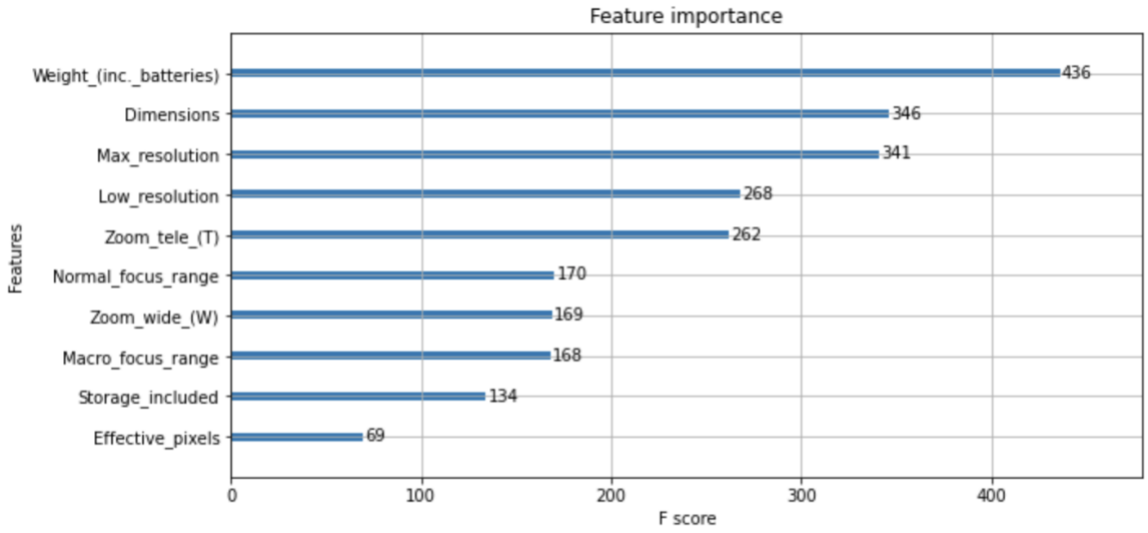
After updating these parameters and training the most optimal model, Best MAE: 214.33 in 27 rounds.

Conclusion/Visualizations

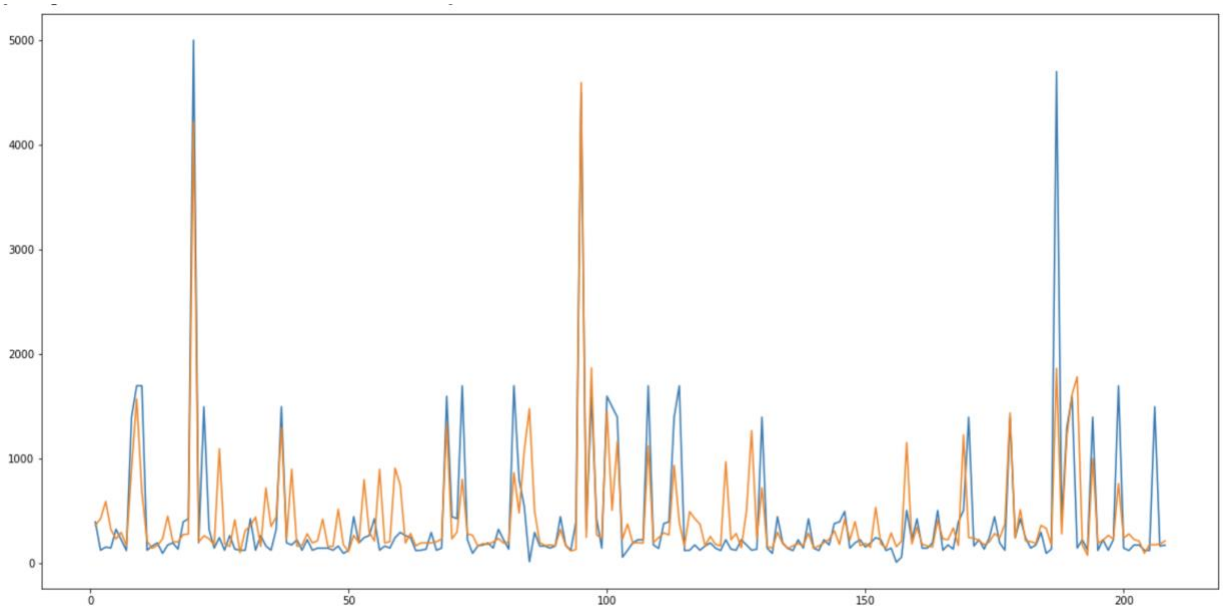
Compared to the baseline, XGB doubled the model's performance, but after optimizing parameters relative to the default, MAE only lowered by around $10. I attribute this $214 MAE to the few data points with abnormally high price tags. I believe that the model could be more precise with a lower range in prices. Also, as I noted earlier, brand and release date are more qualitative influences on camera price that were not included. Below are a few visualizations to better understand the final model



This is a visualization of the depth and node splits of the tree. The particular model represents num_trees = 0.



Here is an update of feature importance. It is much different that the initial graph, which indicates that many of the technical specifications carry varying levels of importance.



Above is a graph of the test data set against the prediction of the test set (orange = preds, blue = test). There are a few major spikes in the test set that aren't fully captured by predictions. In the lower price range, it looks like it is all over the place.

In reflection, I believe that to improve this model, there would need to be a label encoder used on brand and release date to incorporate them into the model. Additionally, I would like to see how removing the outliers affects the predictions of cheaper cameras.