American University of Armenia

Capstone Project

Scalable Data Collection and Machine Learning for Automated Valuation of Armenian Real Estate

Author:
Davit Martirosyan

Supervisor:
Erik Arakelyan

A project submitted in fulfillment of the requirements for the degree of BS in Data Science

in the

Zaven & Sonia Akian College of Science and Engineering

May 9, 2022

Contents

Abstract

Real estate is one of the major sectors of the Armenian economy and has been developing dynamically. Recently, large online platforms have developed in Armenia to advertise real estate offerings, thus reducing information asymmetry, and increasing liquidity in both sales and rental markets. With granular data concerning a representative portion of the real estate offering available online, it is increasingly tenable to monitor the real estate market and develop analytical tools that can accurately estimate the value of real estate assets based on their internal and external features. This research sets out to not only assess the performance of a special class of machine learning models – tree–based bagging and boosting ensembling methods, in estimating the prices of apartments and houses in Armenia, but also create a highly accurate Computer Vision framework, the purpose of which is to correctly predict which of the following design styles a real estate product corresponds to: Modern, classic or soviet. We created scalable data collection pipelines to create an Armenian Real Estate database which is further used to develop robust models for price prediction and interior style detection. Our experiments showed that the performance of XGBoost exceeds that of the Random Forest and Catboost models. Furthermore, using the SHAP approach for feature importance calculation, we have determined that the top three most decisive factors are surface area, coordinates and the material for predicting apartment value and amount of bathrooms, coordinates and interior area for predicting house value. The best results for price prediction were achieved through the XGBoost model, yielding an R-squared of 0.69 for houses, and 0.83 for apartments. We further enhance our understanding of the important features for automated price prediction by assessing various deep learning architectures for visual interior style prediction in a few-shot fine-tuning setting.

Introduction

Real estate is defined as a property, made up of land and any physical structures on it such as various types of buildings. The Armenian real estate market size was valued at $880.4 million in 2018, and is projected to reach $1.25 billion in 2026, growing at a compound annual growth rate of 4.3%. [11].

Increasing transaction volumes and price volatility are the two main motivations for building an automated valuation framework. The latter can help stakeholders in the real estate sector detect potential market opportunities and reduce overhead cost by automating significant portions of the valuation workflow. Furthermore, automated valuation reduces the likelihood of human error or malfeasance negatively impacting the valuation process. The goal of this research is to assess the performance of three different classes of tree – based machine learning ensembling methods, named XGBoost, Catboost and Random Forests, and create a Computer Vision framework that accurately predicts the style of the interior of a real estate product. Related works show that the previously discussed regression methods exceed the performance of other classical machine learning models. Furthermore, their primary advantage over neural networks, which often match them in performance, is that their hyperparameters are more easily tunable and model performance is more explainable.

Related Work

Automated real estate valuation is a popular topic in applied machine learning, and recent work describes the effectiveness of different tree-based ensembling techniques for accurate price prediction. We decide to focus upon the recent advacements in the field and discuss the most prevalent and promising approaches.

In their recent paper, *Prediction and Analysis of Chengdu Housing Rent Based on XGBoost Algorithm* [8], the authors compare the performance of three techniques: LightGBM, XGBoost, and Random Forest Regressor. According to the paper, the most promising performance was obtained by the XGBoost model. Using parameter tuning, the latter attained a coefficient of determination (R-squared) of 0.85 (on a scale of 0 to 1, 1 indicating excellent performance) [8]. Furthermore, they achieved the following results:

Model	MSE	R2
RandomForestRegressor	0.06	0.83
XGBoost	0.04	0.85
LightGBM	0.05	0.84

Another paper, titled *Product marketing prediction based on XGboost and LightGBM algorithm* [7], discusses the LightGBM and XGBoost models for product marketing prediction. The overall conclusion that the paper reached is that they both perform relatively similarly, however, the overall RME's of the XGBoost model is relatively smaller. [7]

Research Methodology

In this section we will describe the stages of this project along with the methods we implemented. The project can be divided into the following main stages: Data collection, data preprocessing, and modeling.

The first stage, as mentioned above was naturally data collection. We created end-to-end data collection pipelines for the following online real estate markets: "list.am", "myrealty.am", "estate.am", "bnakaran.am", and "realestate.am". As we will be describe the data in greater detail in the Data section .

Following data collection we needed to complete the data preprocessing stage. We segmented the data into two different datasets: One containing data that includes information about facilities (whether a given real estate product came with electricity, water, gas, etc.), with a total number of 13025 observations; and the other which did not contain the latter, and had 43504 observations. Our initial hypothesis was that using the dataset which did not contain information about facilities would yield greater performance for the Machine Learning models and as the later analysis proved that we were correct. Next, we concatenated the five separate datasets into one, normalized database. Afterwards, we had to create a mapping function, that mapped elements such as materials into general names (different sources had different names for the same elements, e.g. monolith, Monolit, etc.). We later one-hot-encoded categorical features such as material in order to be able to feed the data to the models. Finally, we implemented outlier detection with the interquartile range (IQR) method for the following features: Price, interior area, room count and floor count. We filtered out any row that had values higher than the IQR multiplied by 1.5 plus the third quartile (Q3), or lower than the first quartile, Q1 subtracted by IQR multiplied by 1.5, for any of the above mentioned columns.

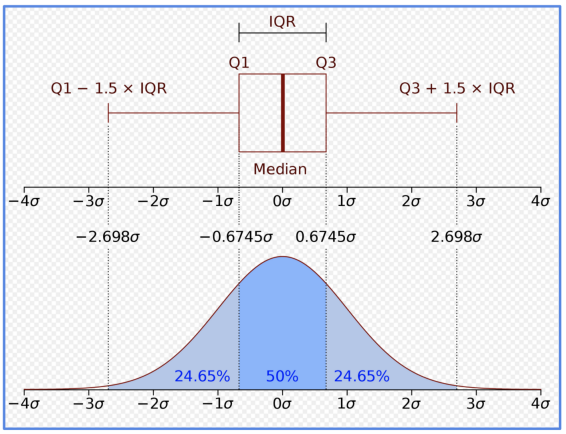

Figure 1: IQR method

In the final stage, we fed the data to various ML models for the price prediction portion of this project, and trained a set of Computer Vision architectures that predict the interior style of the given real estate product, choosing among one of the following: Soviet, classic, or modern. Firstly, we will introduce the Computer Vision portion of this project. As Armenian real estate is quite specific, there was no data available online to be downloaded in order to train such set of models, so we developed a script that scrapes images off of Google. The script simulated human-like behaviour using various engineering solutions, hence we scraped 100 images for each of the three classes without getting blocked by Google. After collecting the data, we designed a parallel data loading pipeline that randomly selects the images and splits them into training and testing sets. Finally, using PyTorch we implemented a transfer learning approach, which is a machine learning method where one reuses a pre-trained model as the starting point or a feature extractor for a model on a new task. We fed the data to the set of the pre-trained models with an addition of a learnable classification head on top, which we fine-tuned in a few-shot setting, while keeping the parameters of the pre-trained model frozen.

Next we split the already pre-processed data into two sections: houses and apartments. This was a necessary step as the data for houses is generally more volatile compared to apartments which is why one model for both real estate types would result in a lower accuracy. Using hyperparameter tuning, we fed the data to three models: XGBoost, Random Forest Regressor, and Catboost. Also, we have used the *Shapley Additive exPlanations* (SHAP) approach to obtain feature importance values for each case.

XGBoost

Extreme Gradient Boosting (XGBoost) [2] is an improved version of the Gradient Boosting Decision Tree (GBDT). This algorithm is composed of multiple decision trees,

and the gradient descent method is used to "boost" each tree, meaning that we learn to adjust the regressor per the error found in a specific tree. Based on all single decision trees, the optimization is carried out by minimizing the loss function as the objective. Unlike the GBDT algorithm, the XGBoost algorithm can automatically use the CPU for multi-threaded parallel computation and carry out Taylor's second-order expansion on the loss function. Meanwhile, the tree model complexity is taken as a regular term in the target function to avoid overfitting. The target function of the XGBoost algorithm is as follows [7]:

$$L(f_t) = \sum_{i=1}^{\infty} l\left(y_i, \hat{y}_i^{t-1}\right) + \Omega(f_t) + C \quad (1)$$

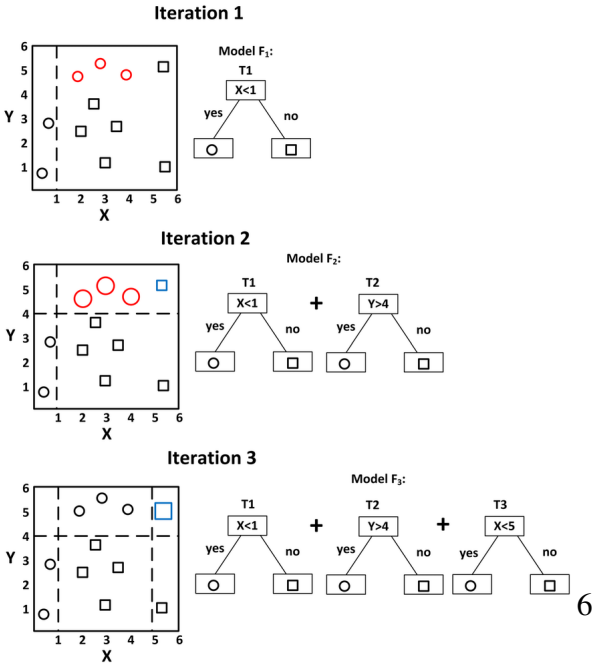

Figure 2: Gradient boosting

Random Forest

The Random Forest Algorithm [1] is composed of multiple decision trees, each with the same nodes, but using different data, leading to different leaves. Merging the decisions of multiple decision trees, the algorithm finds an answer, which represents the average of all these decision trees. When using the Random Forest Algorithm to solve regression problems, the mean squared error (MSE) is used to know how the data branches from each node:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2 \quad (2)$$

Where N is the number of data points, f_i is the value returned by the model and y_i is the actual value for data point i. [1]

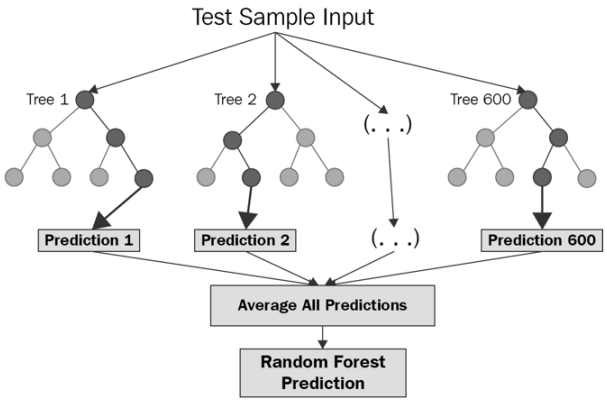

Figure 3: Random forest

Catboost

CatBoost builds upon the theory of decision trees and gradient boosting. The main idea of boosting is to sequentially combine many weak models and through greedy search create a strong competitive predictive model. One of CatBoost's core edges is its ability to integrate a variety of different data types, such as images, audio, or text features into one framework. CatBoost also offers an idiosyncratic way of handling categorical data, requiring a minimum of categorical feature transformation, opposed to the majority of other machine learning algorithms, that cannot handle non-numeric values. From a feature engineering perspective, the transformation from a non-numeric state to numeric values can be a very non-trivial and tedious task, and CatBoost makes this step obsolete. [9]

Computer Vision

To create a scalable and accurate methodology for classifying the interior designs, we use idea of Transfer Learning [15] and Domain Adaptation [14]. In particular we fine-tune a set of object detection and classification models pre-trained on the ImageNet task [10]. Particularly, we use these models as frozen feature extractors and add a learnable dense layer on top for classification.

The models we considered stem from various CNN architectures and span paradigms, ranging from residual connections, knowledge distillation to very deep convolutional architectures. The particular architectures are Resnet18 [3], AlexNet [6], VGG11 [12], SqueezeNet [5], Densenet [4], Inception [13]. These are chosen to test form the best accuracy/speed ratio we are able to obtain throughout the training and inference processes.

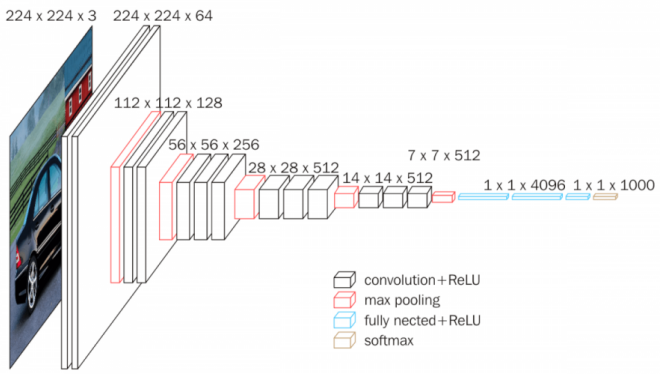

Figure 4: VGG Architecture

Data

In order to acquire the necessary data, we had to design data collection pipelines for the following online real estate websites: "list.am", "myrealty.am", "estate.am", "bnakaran.am", and "realestate.am". After concatenating the data into one database the observation count amounted to a total of 44788 observations, the vast majority of which came from "list.am". Below is additional information about the data collected from each website:

Website	Observations
list	35196
myrealty	5656
bnakaran	1040
realestate	7850
estate	220

The following features were extracted for each real estate product: Room count, interior area, total area (if available, usually valid for houses), apartment floor (which floor the apartment is on), floor count (how many floors a given real estate product has), bathroom count, latitude and longitude, image link (if available), facilities (electricity, water, gas, heating, hot water, internet, air-conditioning, constant water, etc.), the material the structure is made of (monolith, stone, panels, etc.), and the price. The feature designating the facilities will be discussed in more detail in the upcoming sections as after extensive analysis it did not improve model performance. As for the data used in Computer Vision, we designed a robust data collection pipeline that scrapes a desired amount of images from Google without being blocked. We managed to save the images in their original sizes instead of thumbnails provided by Google by carefully engineering the traversal and scraping flows. We scraped 100 images for each of the three classes and later filtered noise from within the images (images that were not exactly related to the query we had made) and completed deduplication.

Data Exploration

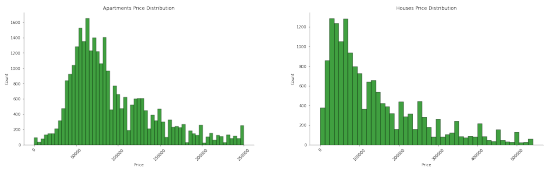

Figure 5: Left: Apartments Price Distribution, Right: Houses Price Distribution

Figure 5 illustrates the distribution of the price of both apartments and houses based on data scraped from the above mentioned sources. This is the target variable that we aim to predict. The histograms peak at around the $60000 to $70000 and $50000 to $60000 price categories for apartments and houses respectively, and decline in reverse proportion to price, as expected.

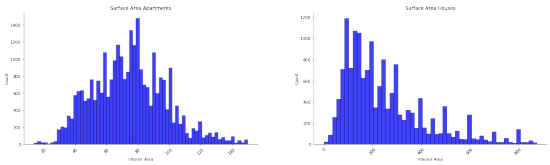

Figure 6: Left: Apartments Area Distribution, Right: Houses Area Distribution

Figure 6 illustrates the distribution of interior surface area. In the case of apartments, we can see a relatively normal distribution with most of the data gathered around the 70 to 80 square meters mark, while for houses the 100 to 200 square meter mark. This is expected, as houses usually tend to have a larger surface area in comparison to apartments.

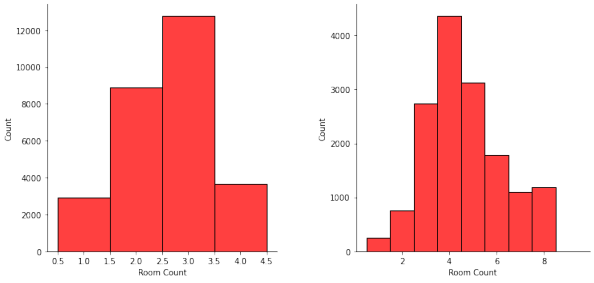

Figure 7: Left: Apartments Room Distribution, Right: Houses Room Distribution

In *Figure 7* can be observed the distributions of rooms. Apartments mostly have 3 rooms, while houses 4 to 5. We can also see that in case of houses the total options for amount of rooms is more than that of apartments.

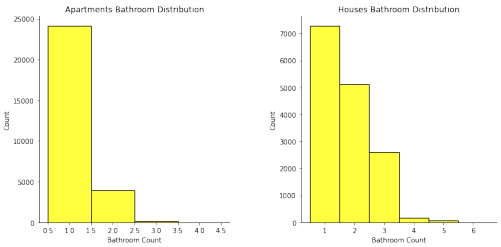

Figure 8: Left: Apartments Bathroom Distribution, Right: Houses Bathroom Distribution

Figure 8 represents the distribution of the amount of bathrooms. Similar to the amount of rooms, houses have more options with the most frequent option being 1 to 2 bathrooms, while in the case of apartments, 1 bathroom.

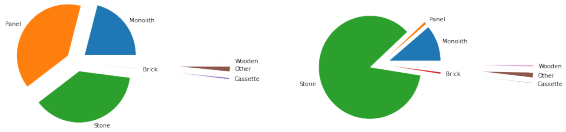

Figure 9: Left: Apartments Materials, Right: Houses Materials

Finally, in *Figure 8* we can observe what materials apartments and houses are mainly constructed with. The top most frequently used materials in case of apartments are panel, stone and monolith with brick and wood and cassette being the least. In case of houses, stone (by far the most frequently used material) and monolith are used most often, with cassette, brick and wood being used the least.

Results & Analysis

Price Prediction

After subsetting the data into houses and apartments, we checked each column for missing values and found that there were very few columns that contained missing values. We then imputed the median of each column (I used the median instead of the mean as this is better when the distribution of a given column is not completely normal). We then split the data into training and testing sets for both datasets (the one that included information about facilities mentioned above, and the one that did not).

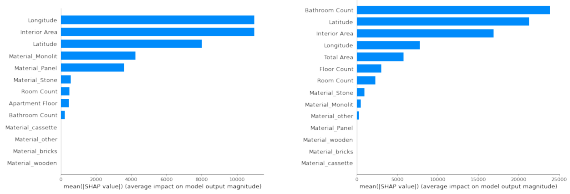

Figure 10: Left: Apartments Important Features W/O Facilities, Right: Houses Important Features W/O Facilities

Feature Importance

As mentioned above we used SHAP for feature importance. We implemented the latter on both datasets, subsetting houses and apartments. In the graphs below can be observed the results:

In *Figure 10* we can see the important features for apartments and houses using the dataset that excludes information regarding facilities (and has around three times more observations). Interestingly, in the case of houses, the bathroom count feature is considered to be the most important feature, whereas in the case of apartments, the significance of the latter is not as high. However, in both cases the interior area and coordinates of a real estate product are highly significant.

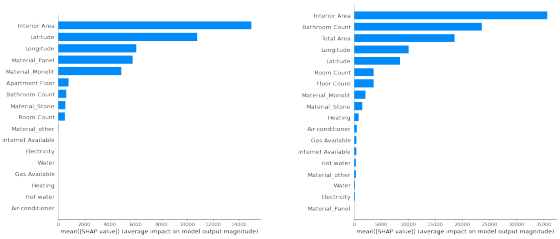

Figure 11: Left: Apartments Important Features with Facilities, Right: Houses Important Features with Facilities

In *Figure 11* are illustrated the same plots, differing only in that the dataset used included information about facilities and had almost three times less observation points. When compared to the previous plot of feature importance for houses, in this case interior area is ranked as the most important feature, where in the previous case it was bathroom count. Furthermore, we can see that for apartments, all facilities are ranked as insignificant, and in the case of houses, they are ranked as least significant.

Regression Models

In this section, we will show the results attained using both datasets. As already noted, all models underwent hyperparameter tuning in order to get the optimal parameters for each. The evaluation metric mean absolute error (MAE) shows the absolute value of the average error (how far the model predicted from the actual value). Firstly, we will show the results from the dataset where information regarding facilities is not included:

Model Houses	MAE	R2
RandomForestRegressor	46423	0.65
RandomForestRegressor with Feature Importance	46424	0.65
XGBoost	43173	0.69
XGBoost with Feature Importance	43209	0.69
Catboost	48983	0.63
Catboost with Feature Importance	49014	0.63

From the table above, it can be observed that the best results for the houses model was yielded by the XGBoost algorithm, achieving an R-squared of 0.699, and an MAE of 43173.

Model Apartments	MAE	R2
RandomForestRegressor	13599	0.80
RandomForestRegressor with Feature Importance	13597	0.80
XGBoost	12579	0.83
XGBoost with Feature Importance	12647	0.83
Catboost	15703	0.76
Catboost with Feature Importance	15733	0.76

The table above shows the results for the apartments model. Similar to the houses model (yet with a much higher accuracy, as expected), the XGBoost model again yields the highest accuracy with an R-squared of 0.83 and an MAE of 12579.

Next let us examine the regression models' performance on the dataset which included information about facilities:

Model Houses	MAE	R2
RandomForestRegressor	72155	0.56
XGBoost	59730	0.64
Catboost	76689	0.53

From the table above it is evident that all models performed worse using this dataset. The best model is again the XGBoost model, yielding an R-squared of 0.64, and an MAE of 59730.

Model Apartments	MAE	R2
RandomForestRegressor	19303	0.76
XGBoost	17177	0.79
Catboost	21565	0.72

Similar to the table above, all models performed worse. The best model is again the XGBoost model, yielding an R-squared of 0.79, and an MAE of 17177.

Hence, we can conclude that the dataset excluding information about facilities results in models that perform better. This may be because of the fact that it contains more observations, however, the feature importance models also indicate that facilities are not important features. Also, the XGBoost model proved to outperform the other models in all cases.

Residuals Visualizations

Finally, some visualization that will help better understand model performance:

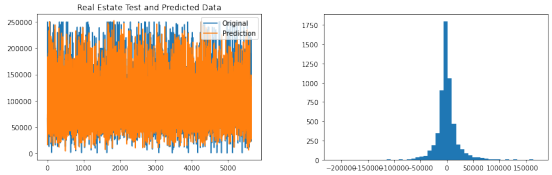

Figure 12: Left: Actual vs Prediction Apartments, Right: Residuals Distribution Apartments

The right most visualization in *Figure 12* illustrates the distribution of the residuals for the best apartments model. As we can see, it shows a normal distribution, indicating that the model performs well.

Computer Vision Model

In this section we will talk about the results that we got from training the Computer Vision model, which had to predict if the interior design of a given real estate product corresponds to a soviet, classic or modern style. We tried several experimental settings with various model architectures presented in the previous section .

Experiment_name	Class_id	Precison	Recall	F1
SqueezeNet batch = 4	Classic	0.78	0.78	0.78
	Modern	0.65	0.68	0.67
	Soviet	0.73	0.70	0.71
SqueezeNet batch = 8	Classic	0.70	0.62	0.66
	Modern	0.70	0.83	0.76
	Soviet	0.68	0.64	0.66
SqueezeNet batch = 16	Classic	0.91	0.73	0.81
	Modern	0.69	0.67	0.68
	Soviet	0.69	0.84	0.76
Resnet batch = 4	Classic	0.72	0.78	0.75
	Modern	0.44	0.50	0.47
	Soviet	0.47	0.39	0.43
Resnet batch = 8	Classic	0.82	0.72	0.77
	Modern	0.76	0.80	0.78
	Soviet	0.80	0.85	0.82
Resnet batch = 16	Classic	0.73	0.61	0.67
	Modern	0.56	0.89	0.69
	Soviet	0.73	0.50	0.59
VGG batch = 4	Classic	0.74	0.67	0.70
	Modern	0.53	0.67	0.59
	Soviet	0.77	0.71	0.74
VGG batch = 8	Classic	0.76	0.71	0.74
	Modern	0.72	0.90	0.80
	Soviet	0.74	0.54	0.62
VGG batch = 16	Classic	0.77	0.67	0.7
	Modern	0.83	0.90	0.86
	Soviet	0.76	0.76	0.76

Alexnet batch = 4	Classic	0.76	0.62	0.68
	Modern	0.84	0.84	0.84
	Soviet	0.67	0.80	0.73
Alexnet batch = 8	Classic	0.83	0.81	0.82
	Modern	0.80	0.80	0.80
	Soviet	0.86	0.88	0.87
Alexnet batch = 16	Classic	0.76	0.70	0.73
	Modern	0.79	0.82	0.81
	Soviet	0.72	0.76	0.74
Inception batch = 4	Classic	0.43	0.56	0.49
	Modern	0.52	0.67	0.59
	Soviet	0.75	0.46	0.57
Inception batch = 8	Classic	0.60	0.79	0.68
	Modern	0.66	0.78	0.71
	Soviet	0.74	0.40	0.52
Inception batch = 16	Classic	0.66	0.59	0.62
	Modern	0.67	0.77	0.72
	Soviet	0.56	0.51	0.54
Densenet batch = 4	Classic	0.59	0.56	0.57
	Modern	0.59	0.73	0.65
	Soviet	0.62	0.50	0.56
Densenet batch = 8	Classic	0.83	0.60	0.69
	Modern	0.69	0.78	0.73
	Soviet	0.60	0.71	0.65
Densenet batch = 16	Classic	0.96	0.62	0.76
	Modern	0.79	0.80	0.80
	Soviet	0.63	0.85	0.73

Table 1: Complete results through various models and experimental settings.

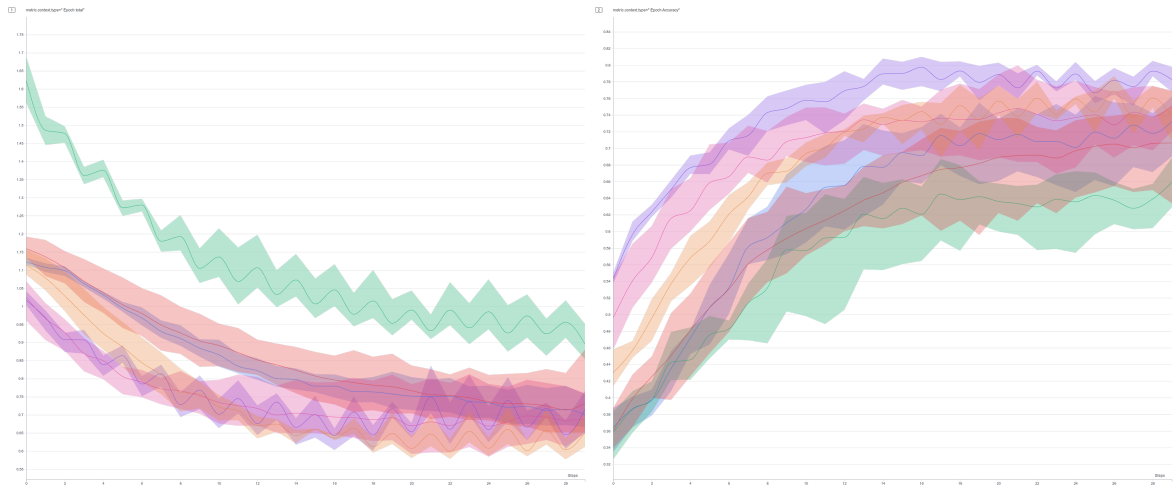

Figure 13: Averaged experiments per model.
Left: Loss Aggregated per model architecture (Averaged Per Run).
Right: Accuracy aggregated per model architecture (Averaged Per Run)

We also test against a vast hidden test set in order to validate our few-shot fine-tuning approach. Through our experimentation we find that the most efficient architectures in terms of the performance on the hidden test are VGG and AlexNet, however in terms of training and inference speed SqueezeNet is able to outperform the other models with only a small drop in the reported metrics. All of the reported numbers are an averaged out version per the experimental setting where we keep only the set of most successful runs. We considered a search along various hyper-parameters per experiment, i.e., the learning rate, momentum, batch size and embedding dimensionality. We use Aim for experimentation tracking and make the complete set of experiments available in our repo.

Discussion & Conclusions

We have assessed the performance of three families of tree – based ensemblers and determined that XGBoost performs better than Random Forest and Catboost on data extracted from the Armenian real estate market. In collecting the data, we have also developed a stable and scalable data collection pipeline that can continuously increase the size of the dataset and iteratively improve model performance. We also developed a framework for few-shot Deep CNN fine-tuning that was used for visual interior style detection. We conducted a thorough research across model architectures and hyperparameters and showed better models in terms of performance and speed. In the future, we plan on integrating the output of the Computer Vision framework as a feature for the price prediction models, potentially enhancing the latters' performances.

Bibliography

[1] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[2] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[5] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[7] Y. Liang, J. Wu, W. Wang, Y. Cao, B. Zhong, Z. Chen, and Z. Li. Product marketing prediction based on xgboost and lightgbm algorithm. *Proceedings of the 2nd International Conference on Artificial Intelligence and Pattern Recognition - AIPR '19*, 2019.

[8] Y. Ming, J. Zhang, J. Qi, T. Liao, M. Wang, and L. Zhang. Prediction and analysis

of chengdu housing rent based on xgboost algorithm. *Proceedings of the 2020 3rd International Conference on Big Data Technologies*, 2020.

[9] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[11] R. Sawsnt and O. Sumant. Armenia real estate market size, share: Forecast and analysis - 2026. https://www.alliedmarketresearch.com/armenia-real-estate-market-A06057#:~:text=A.-,The%20Armenia%20real%20estate%20market%20size%20was%20valued%20at%20%24880.4,4.3%25%20from%202019%20to%202026., 2020.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[14] M. Wang and W. Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.

[15] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.