Case Study

University of Zurich

Department of Banking and Finance

FT 2023

Supervisor: Benjamin Zimmermann

Optimizing Real Estate Price Class Estimation: An Exploration of Machine Learning Techniques

Victor Nicolae Simtion -22-714-497- victornicolae.simtion@uzh.ch

Lev Maznichenko - 21-751-821 - lev.maznichenko@uzh.ch

Minh Hien Tran - 16-918-732 - minhhien.tran@uzh.ch

Anne-Catherine Sophia Walder - 21-722-475 - anne-catherinesophia.walder@uzh.ch

Zurich, 16th April 2023

Table of Contents

1. List of Figures and Tables

2. List of Abbreviations

KNN	K-Nearest Neighbour
LDA	Linear Discriminant Analysis
QDA	Quadrant Discriminant Analysis
SVM	Support Vector Machines

1. Abstract

In this case study, we have explored the application of machine learning algorithms to predict real estate price classes using a dataset of residential homes in Ames, Iowa. We have built four different models, one implementing K-Nearest-Neighbours (KNN), one implementing Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), another Random Forest Classifier and one more implementing Support Vector Machines (SVM), which turned out to be the most performant combined with Random Forest Classifier.

2. Methods

In this section we will briefly elaborate on our four different models we used as a first approach to this case study.

2.1 KNN by Victor

Victor explored different techniques to improve the system, such as feature scaling and feature selection. He tested MinMaxScaler and RobustScaler, and compared the results with the StandardScaler. The best scaling for the model turned out to be the StandardScaler. Victor experimented with different feature selection methods, including SelectPercentile, SelectKBest, SelectFromModel, and Recursive Feature Elimination (RFE). The highest test score achieved was 0.82 using the SelectKBest feature selection. The trade-off between accuracy and performance was best suited to the Randomized Search classifier. To address potential class imbalance issues, Victor tried stratified sampling for the train-test split and StratifiedKFold cross-validation. However, the results were inconclusive, and the gap

between the KNN model score and test results did not consistently reduce.

2.2 LDA and QDA by Minh

Minh started with data pre processing steps such as removing columns with too many values, replacing non-numeric values with NaN and replacing NaNs with zeros. Then he applied feature scaling using StandardScaler, and class imbalance correction using RandomOverSampler. Next, Minh splitted the resampled data into training and testing sets, performs hyperparameter tuning using GridSearchCV, and selects the best hyperparameters for LDA and QDA models separately. Then, he fitted the LDA and QDA models with the best hyperparameters, makes predictions on the testing set, and evaluates the performance of the models using accuracy, confusion matrix, precision, recall, F1 score and ROC-AUC (Receiver Operating Characteristic - Area Under Curve). The accuracy he received for LDA and QDA were 0.8 and 0.83 respectively.

2.3 Random Forest Classifier by Anne-Catherine

Anne-Catherine started by encoding categorical variables and dealing with missing values by analyzing the nature of the parameter with missing values. After splitting the data into training and testing sets, she implemented an instance of the Random Forest classifier. She defined outer and inner cross-validation schemes and a range of hyperparameters for the search space. RandomizedSearchCV was used to perform the grid search and fit the model. The best CV accuracy was found to be 0.83, and the best parameters were as follows:

- n_estimators: 560

- max_features: auto
- max_depth: None
- min_samples_split: 5
- min_samples_leaf: 1
- bootstrap: False

The model's performance was evaluated using accuracy (0.84), precision (0.83), recall (0.84) and f1 score (0.83). The results indicated that the Random Forest Classifier model was effective in predicting real estate price classes.

2.4 Random Forest Classifier after modification by Lev

Lev started by trying to implement the SVM algorithm. After it was obvious that its accuracy is small and that Random Forest's accuracy without data processing (handling outliers/class imbalance) exceeds the accuracy of all other algorithms AFTER processing, and knowing that it works well on models with small amounts of data, the decision was made to choose it.

Lev decided to handle missing numerical values with median and missing categorical values with frequent (it showed the best accuracy). He tried to play with outliers and class imbalance, but it hardly affected the accuracy or did not at all. He then defined the first version of hyperparameters and it showed quite good results. Subsequently he applied quite calculation-expensive hyperparameters and since he had only one attempt, he decided not to take the risk and perform a model without outliers and class imbalance but with cross-validation of 10. The resulting accuracy is 0.90 with following hyperparameters:

- n_estimators: 200
- max_features: sqrt
- max_depth: None
- min_samples_split: 5
- min_samples_leaf: 2
- bootstrap: False

3. Discussion

In the following paragraphs we will explain how we handled different challenges of the data set and present our final results.

3.1 Missing Data and outliers

We inserted the median for numerical data and most frequent for categorical. In the final version of code we did not handle any outliers, since in previous versions it did not affect the accuracy or just minimally.

3.2 Class Imbalance

Class imbalance occurs when the number of instances in each class is not equal. This can lead to biased models that perform well on the majority class but poorly on the minority class. In our code, class imbalance is handled by setting the 'class_weight' parameter of the RandomForestClassifier to 'balanced'. This adjusts the weights of each class to be inversely proportional to their frequency, which helps to balance the impact of each class on the model's performance. Additionally, other techniques such as oversampling or undersampling can be used to balance the classes.

4. Results

With our final code these were our results:

	precision	recall	f1-score	support
0.0	0.94	0.76	0.84	21
1.0	0.94	0.96	0.95	194
2.0	0.83	0.84	0.84	58
3.0	0.53	0.62	0.57	13
4.0	0.67	0.33	0.44	6
accuracy			0.90	292
macro avg	0.78	0.70	0.73	292
weighted avg	0.90	0.90	0.90	292

To compare we have compiled two graphs that show the results for our predecessor models as well as our final model.
The first one shows the accuracy of the different models in predicting the classes. Due to problems with the python library we were unable to compile the graph for LDA and QDA. Regardless it shows consistently that class 1 was predicted the best by all models whereas class 4 was predicted the worst overall. This is due to class imbalance as discussed previously.
The second graph shows the weighted average precision and F1-score of the models. As is clearly visible our final code reached the highest score. From the predecessor models Random Forest scored the highest, which was also the deciding factor on continuing with the model for our final code.

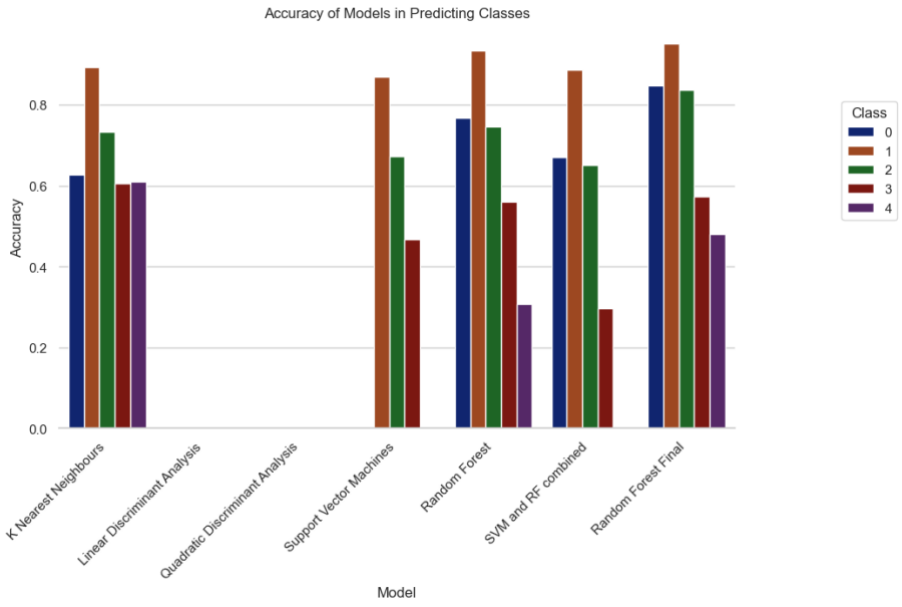

Figure 1: *Accuracy of Models in Predicting Classes*

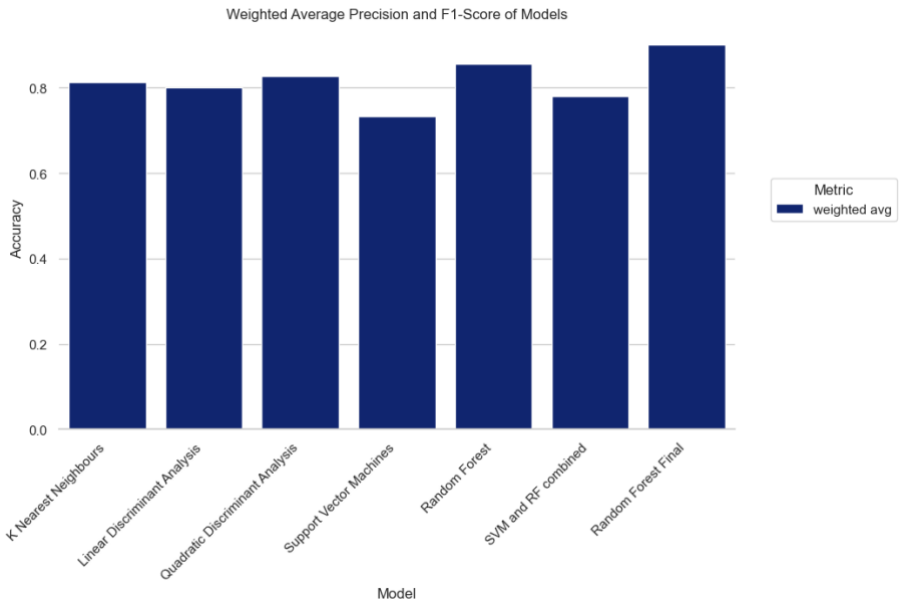

Figure 2: *Weighted Average Precision and F1-Score of Models*

Hereby we declare that the presented paper is all our own work and was produced without the use of other than the stated resources. All passages that are directly or indirectly taken from published or unpublished texts are marked as such. This paper has not been presented in the same or a similar form or in extracts in the context of another exam.

Zurich, ... Signature