Machine Learning in Finance 2023

Cyrill Stoll, Arthur Schlegel, Aleksandar Kuljanin, Selina Waber

April 2023

1 Data Set

The Ames Housing dataset was compiled by Dean De Cock for use in data science education. The data set is describing the sale of individual residential property in Ames, Iowa from 2006 to 2010. The data set contains 1460 observations and 81 explanatory variables. We have 43 categorical variables and 37 numerical variables. One is the target variable "Class" which indicates the price class of the house. Another one is an ID-variable that has no predictive power. Therefore, 79 variables are descriptive variables that should explain Class.

2 EDA

After loading the data set, we started with an initial EDA to get an overview of the data at hand. This initial EDA should help us to identify the takeaways and possible problems we should look out for. Firstly, the data set contains a lot of qualitative variables. Nearly all but two of the qualitative variables are not binary. Meaning that almost all of them have more than one possible option. Secondly, another problem that we saw is that there are variables that are numerical but describe a qualitative feature and thus can be interpreted falsely by the models.

However, the biggest problem was that there are a lot of missing values for a number of variables. For some of the variables, more than half of the values are missing. We addressed these problems in the missing data handling and feature engineering.

2.1 Missing Data

Out of the 79 initial variables, 19 have missing values. This goes from a single missing value in the variable "Electrical" to over 90% missing values in "Pool", "MiscFeature" and "Alley". Variables with so many missing values pose a huge problem for our models and are also very difficult to handle.

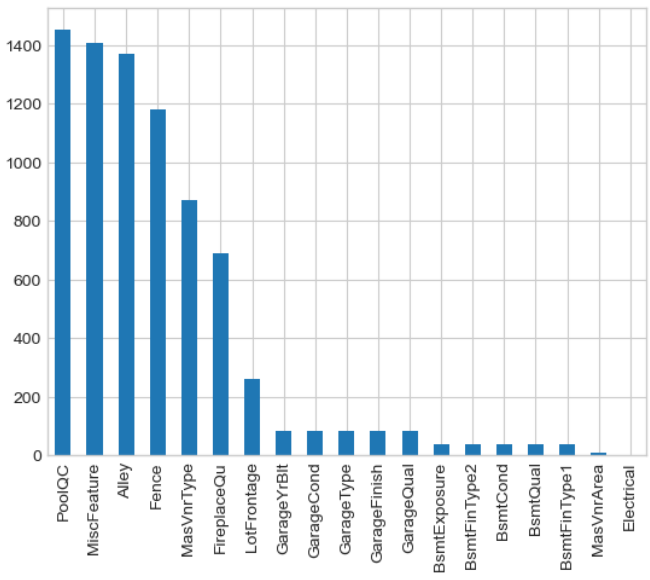


Figure 1: Missing Data

To understand why so many values are missing for these variables we consulted the data description sheet. In the data description, it is mentioned that for many variables NA does not necessarily mean the value is missing. For most of our variables, NA means that the specified trait of this variable is non-existent. NA in the Alley variable for example means that there is "no alley access".

The following variables have NAs that indicate that the trait is non-existent (meaning of *NA* in brackets):

PoolQC (No Pool), MiscFeature (None), Alley (No alley access), Fence (No Fence), FireplaceQu (No Fireplace), GarageCond (No Garage), GarageType (No Garage), GarageFinish (No Garage), GarageQual (No Garage), BsmtFinType2 (No Basement), BsmtExposure (No Basement), BsmtQual (No Basement), BsmtCond (No Basement), BsmtFinType1 (No Basement).

For the above variables we replaced all NAs with the string "No" to indicate that for these observations the specified trait is missing. After filling in these variables,

we are left with only 5 variables that still have some missing values left.

We were quite creative in filling numerical values to some special variables: "GargeYrBlt" where we just assumed that the garage was built in the same year as the house, "LotFrontage" where we were assuming that the mean would be a good approach for missing values or "Electrical" where we were filling in the most frequent type.

This leaves us with a data set that has no longer any missing values.

2.2 Feature Engineering

Further we needed to encode the categorical variables and split the data set into a training and a test set. Firstly, we identified numerical variables that should be handled as categorical ones. Namely this is the case for "MSSubClass" and "MoSold". We then changed them into categorical variables. This had to be done first so they will be included in the encoding. We assigned columns to the feature matrix X and the response vector y. On the feature matrix X, we factorized the categorical variables with the `get_dummies` function, creating (multiple) dummy variables for a categorical variable. The output was assigned to X again. The only exemptions were the two variables "Street" and "CentralAir". The `get_dummies` function would have created two separate dummy variables for each of them (e.g., `CentralAir_y` and "`CentralAir_n`). In case that this would cause a problem with multicollinearity we decided to create a dummy for both variables with the .factorize() function. For the partitioning of the data set into a training and test set a 70/30 (training/testing) splitting was chosen. For reproducibility we use "`random_state` = 42", for further information why the number 42 is used please consult (insert the link to the website). It is a stratified sample so that the proportions of values as in the original data set is maintained.

Further we needed to encode the categorical variables and split the data set into a training and a test set. Firstly, we identified numerical variables that should be handled as categorical ones. We then changed them into categorical variables. This had to be done first so they will be included in the encoding. We assigned the data columns to the feature matrix X and the response vector y. On the feature matrix X, we factorized the categorical variables with the `get_dummies` function, creating (multiple) dummy variables for a categorical variable. The output was assigned to X again. The only exemptions were the two variables "Street" and "CentralAir". The `get_dummies` function would have created two separate dummy variables for each of them (e.g., "`CentralAir_y`" and "`CentralAir_n`"). In case that this would cause a problem with multicollinearity we decided to create a dummy for both variables with the .factorize() function.

For the partitioning of the data set into a training and test set a 70/30 (training/testing) splitting was chosen. For reproducibility we use "`random_state` = 42", for further information why the number 42 is used please consult explanation for the choice of the number 42. We decided to use a stratified sample so that the proportions of values stay the same as in the original data set.

2.3 Creating New Variables and Deleting Old Ones

We created a couple of new variables which we thought will better describe the "Class"-Variable. We summed up the different floor sizes e.g. 'TotalBsmtSF', '1stFlrSF' and '2ndFlrSF' to the new variable 'totalSqFeet'. Further we createt the new variables 'totalBathroom', 'houseAge', 'reModeled' and 'isNew'. We erased the old variables: 'TotalBsmtSF','1stFlrSF', '2ndFlrSF', "FullBath", "BsmtFullBath", "HalfBath", "BsmtHalfBath", "YearBuilt", "YearRemodAdd". We did that in reference to Tran, 2023.

2.4 Outliers

Since the target variable is categorized in bins, identifying outliers is harder. The last bin is for houses with a price over $400'000. So, we don't know if the houses in category 4 are worth $400'000 or much more. We still identified a few outliers. This partially improved the models slightly.

2.5 Class Imbalance

Class imbalance occurs when there is an uneven distribution in a dataset of the classes to be used for classification by the model. Such an imbalance can lead to a poor model, as the model usually achieves higher accuracy if it predicts the more common class and ignores the rare class. As illustrated in the graph below, class 1 is over-represented. To address this issue, on the one hand, SMOTE (Synthetic Minority Oversampling Technique) can be used. This was specifically applied in the KNN and LDA model, but not with the desired effect. The model

did not improve. The overall classification report shows lower precision and recall scores for some classes. This could be due to overfitting on the training data or other factors. Furthermore, random oversampling was used to improve the model. The drop in accuracy could be due to the fact that the synthetic samples generated by Random Oversampling can add noise to the data, leading to overfitting.

3 Models

3.1 Logisitc Regression

By using one or more predictor variables, logistic regression models the likelihood of a binary or categorical outcome. The purpose of this kind of supervised learning algorithm is to predict the class or category of a given sample based on the values of its features. By estimating the logistic function's parameters, logistic regression may convert input variables into probabilities between 0 and 1. The final forecast is then made by thresholding the logistic function's output.

To enhance the performance of our model, we experimented with various preprocessing strategies, hyperparameter tweaking techniques, and feature selection strategies: We applied the StandardScaler and MinMaxScalar preprocessing methods. We obtained a f1 score of 0.78 for MinMaxScalar and 0.76 for StandardScaler. MinMaxScalar increased to a f1 score of 0.81 with hyperparameter adjustment, while StandardScaler increased to 0.82. Techniques for feature selection were also tested. We obtained a f1 score of 0.83 using RandomForest feature selection and StandardScaler feature scaling. The f1 score stayed at 0.82 when we combined the feature selection approach with hyperparameter adjustment. Additionally, we tried both PCA and XGBoost. The f1 score obtained using StandardScaler and XGBoost was 0.80. PCA combined with StandardScaler, on the other hand, produced a f1 score of 0.75.

Overall, our findings indicate that, with a f1 score of 0.83, employing StandardScaler with RandomForest feature selection is the best method for creating a logistic regression model for multinomial data.

3.2 KNN

K-Nearest Neighbors (KNN) is an algorithm used for classification. In this algorithm, the prediction for a new data point is calculated based on the k nearest neighbours from the training dataset. The number of k affects the performance of the algorithm, and in our case the optimal number was determined to be 20. The model achieved an F1 score of 0.75. However, the classes in the data are not uniformly distributed, so SMOTE was applied with an F1 score of 0.67 to adjust for this. Nevertheless, the model did not achieve a satisfactory F1 score and other approaches such as random oversampling with an F1 score of 0.66 were tested.

3.3 LDA and QDA

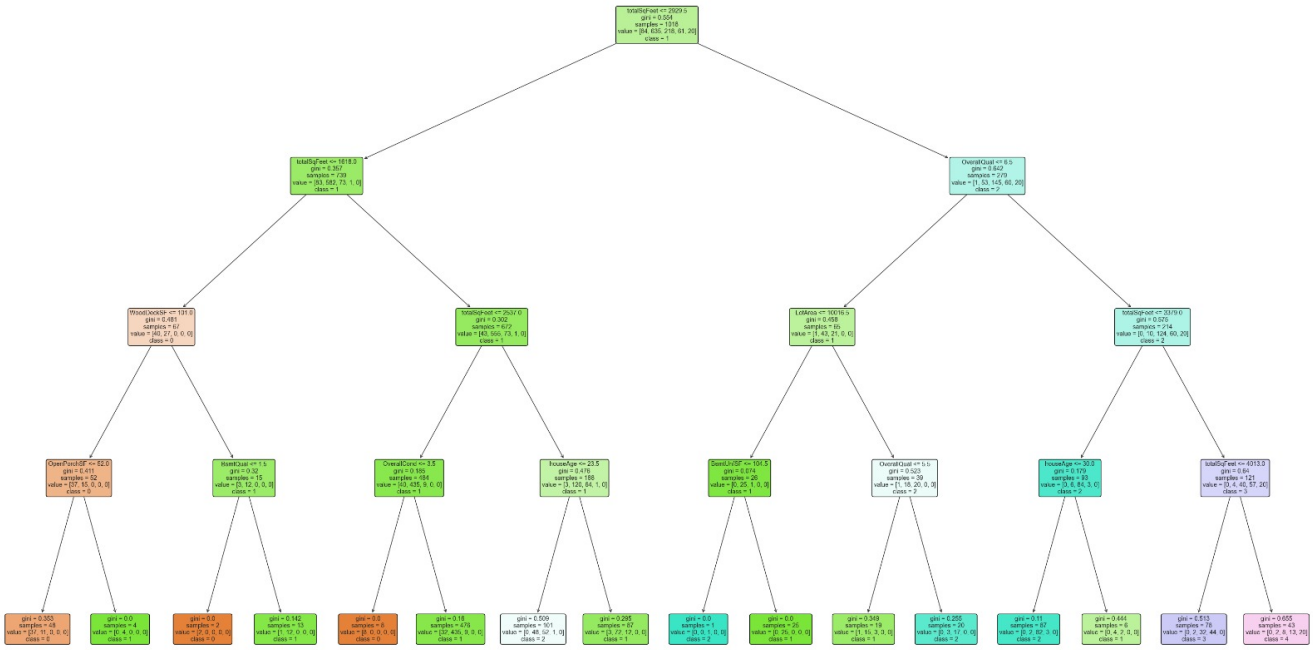
Linear Discriminant Analysis as well as Quadratic Discriminant Analysis try to find a decision boundary between different classes by calculating probabilities and producing estimates for the probabilities of new data points. LDA assumes that the distributions of the different classes are the same and that a linear decision boundary is sufficient. QDA, on the other hand, assumes different distributions for the classes and uses a non-linear decision boundary. In our models, we used the feature variables with the highest correlation to the target variable and an appropriate F1 score of 0.79 using LDA and an F1 score of 0.82 using QDA.

3.4 Decision Trees and Random Forest

The models of decision trees and random forests are very promising since they are invariant to the feature's scale and have a straightforward, hands-down approach. As expected, the decision tree performs quite well without any extensive hyperparameter tuning. The decision tree depicted below shows what features were used. But the test score is quite a bit lower than the training score. This is following the literature because decision trees often suffer from overfitting due to complex decision boundaries.

To address this problem of overfitting, we also conducted machine learning with "Random Forest"-algorithms. With the random forest running multiple decision trees allows us to see which classification is the most probable. The observation is classified according to the majority vote of all the decision trees. The performance is significantly better than for the decision tree. We achieved a performance of up to 0.84. We used hyperparameter-tuning for both the decision tree and the random forest to improve our results. Unfortunately, depending on the number of parameters and values to be investigated, GridSearchCV needs much computational power and thus leads to a

Figure 2: Decision Tree



significant time effort. Therefore, we could not explore all possible values for every hyperparameter. In the final code, we run the GridSearchCV with a preselected number of values. We started with a broad approach and tried to reduce the number of values by closing in on the ideal one. Overall, the random forest and the decision tree performed exceptionally well. Further hyperparameter-tuning might even increase the performance of these models.

3.5 Support Vector Machines

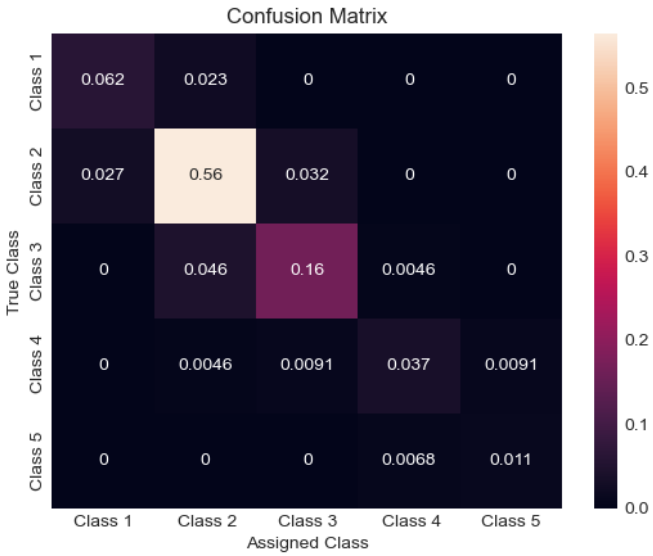


Figure 3: Confusion Matrix with RF Feature Selection

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

Support Vector Machine algorithms are not scale invariant, so it is highly recommended to scale your data. This can be done easily by using a Pipeline. SVM has hyper-parameters such as C or gamma values that can greatly impact its performance. However, determining the optimal hyper-parameters can be a difficult task. One way to do so is by exploring various combinations of hyper-parameters to identify the ones that work best. Often one uses Gridsearch, which involves creating a grid of hyper-parameters and trying out all possible combinations to determine the optimal set of values.

For SVM we did further a feature selection with Random Forests and XG-Boost as well as a dimension reduction with PCA Feature Selection. We got the following results: regarding all features we got a F1-Score of 0.80, for Random Forest feature selection we got 0.83, Xg-Boost we got 0.81 and with PCA only 0.65.

We see that a previous feature selection with Ran-

dom Forests improves the accuracy/F1-Score.

4 Performance Summary

	All Features	RF Selection	XgBoost Selection	PCA
LDA	0.79			
QDA	0.82			
KNN	0.78			
Logistic Regression	0.82	0.82	0.80	0.75
Decision Tree	0.75			
Random Forest	0.84			
SVM	0.8	0.83	0.81	0.65

5 Further Research and Improvements

We have already tested some models that look promising. Nevertheless, there is still room for improvement. PCA is certainly a very good approach for further development and can be used to create new variables. In addition, other performance metrics can be calculated and more hyperparameter-tuning could lead to slightly better results. Another possibility would be neural networks, which use a layer of neurons that provides a probability distribution over the possible classes as output. Finally, we acknowledge that there is room for a more careful examination of the dataset. For example, one could investigate whether some features are linearly dependent or set a stricter threshold for NaN values than the one we applied.

6 Conclusion

The biggest challenge was the data handling. Dealing with a lot of missing data, a strong imbalance in the target variable and in general many variables compared to the number observations.

References

Tran, C. (2023, April 16). *Predict ames house price - advanced regression techniques.* https://chriskhanhtran.github.io/minimal-portfolio/projects/ames-house-price.html (accessed: 16.04.2023)