Feature engineering and its impact on advanced linear models

PATRICK KLAUS BAGINSKI*

CORNELL UNIVERSITY – CORNELL TECH – Contact: pkb42@cornell.edu

May, 16th, 2017 – Using the Kaggle "House Prices: Advanced Regression Techniques" Dataset modified under Apache Spark 2.1.0 (PySpark) to build and exemplify different machine learning feature engineering techniques

ABSTRACT: This paper covers the application of creative feature engineering techniques and their impact on the predictive performance of advanced linear models. Specifically, it uses the Kaggle "House Prices" data set and Apache Spark to build machine learning models and compare their performance based on different feature engineering techniques.

1 Introduction

The Kaggle House Prices: Advanced Regression Techniques data set is a unique compilation of information on the Ames, Iowa housing market, compiled by Dean De Cook and particularly aimed at data science education. The related Kaggle competition [4] itself asks the aspiring data scientists to predict the final price of houses by creatively developing advanced linear regression models and evaluate their performance based on the Root-Mean-Square-Error metric[4]. This classic machine learning application is good at generalizing on the data but given its inherent biases can easily lead to over-generalization. The training set related to the housing market consists of 1460 instances spanning across 79 explanatory features that describe a multitude of aspects regarding residential homes in Ames. The competition stresses to use creative feature engineering to come up with an accurate prediction. To process this task and predict the final house price, the use of distributed and parallelized processing and storage through the means of Apache Spark is applied to the problem space. This is fueled by a cluster of AWS EC2 Instances and the Databricks notebook service.

This paper is structured based on the typical data science process as suggested by Farcaster [5]: The second section describes the data processing of the data, the third section describes the machine learning process and feature engineering techniques. In the fourth section, it addresses the advanced linear models chosen for the experiment at hand as well as the chosen feature engineering techniques. In the fifth section the paper describes the machine learning experiment setup and in the last section compares and interprets the results of this process.

2 Processing the dataset

The data set contains 79 explanatory features regarding the Ames housing market and one predictor variable which is the price for which the house of that instance was sold. Overall, the dataset holistically describes the housing market up to a detail beyond immediate use for machine learning. Many other projects have been conducted based on the data and collectively conclude that only few of the 79 features are truly impactful for the prediction [2,3]. During preprocessing of the data set it becomes obvious that some features have missing values. The way to deal with missing values is a feature engineering class in itself and will be addressed further in the next section. However, given the extensive research on the topic and on the dataset the choice has been made to construct an experiment with a selection of 7 features. These 7 features are according to the highest performing kernels on the Kaggle competition the most impactful ones for advanced linear models. The reason why many competitors chose a narrow selection of features for modelling lies in the assumption that many of the features are correlated amongst each other and a few main features combine the predictive power of other features [2]. This includes a selection of categorical and numerical features. The features chosen and their purpose is listed below:

Feature	Description
SalePrice	The price the house was sold for
OverallQual (numeri-	A variable describing the overall quality of the

cal)	asset for sale
YearBuilt (numerical)	The year in which the house was built
TotalBsmtSF (numerical)	The total square footage of the basement of the house (0 if no basement)
GrLivArea (numerical)	Above grade (ground) living area square feet
Street (categorical)	The type of road access the house has
Neighborhood (categorical)	Physical location within Ames city limits
HouseStyle (categorical)	Type of dwelling

3 Machine learning process and "The Lego Model for Machine Learning"

To properly address the ask of creative feature engineering a quick recap on "The Lego Model for Machine Learning" will be provided [1]. In professional and business applications, the data science process is being automated including the step of machine learning. The general process that is being automated includes 11 steps in the broader sense (there are more in the original Lego Model for Machine Learning):

1. Feature Classification – A process in which features are classified as numerical, binary, categorical, …
2. Deriving features – After classification, features are derived and chosen for the modeling exercise
3. Missing values – The process of addressing missing values in the data set, e.g. using mean of the feature to replace missing values or mode (for categorical features)
4. Feature transformation – This process includes steps such as normalizing, bucketing and scaling features
5. Augmenting features – This technique uses existing features in a data set to look up additional information from other datasets relating to this feature, e.g. having "City" as a feature can be used to add "average income" or "crime rate" to the dataset
6. Interaction features – Interaction features are concatenations or arithmetic combinations of existing features in the data set, e.g. Gender multiplied by Age
7. Label Leakage – A process where features are tested for their correlation with and interaction with other features, screening for biases
8. Sanity Checking – A process in which the data set is tested for inconsistencies and statistically undesired interactions between features and instances
9. Re-sampling – A process in which the data set is re-sampled to achieve a statistically better distribution of instances and features
10. Model selection – The stage in which the data set is fed into a multitude of models to test for performance and select the best models
11. Hyper Parameter Tuning – A process in which a given model is getting tested on different parameters to achieve better performance

4 Feature Engineering and advanced linear models

To limit the vast space of feature engineering techniques a choice of six feature engineering techniques has been addressed for this paper. These feature engineering techniques are applied to the training data set using advanced linear machine learning models. In broad terms, the family of linear models has been chosen due to its good performance in generalizing across the data as well as to predict accurate sales prices for the houses in Ames. Linear models are generally more prone to bias and the present case is no exemption. To overcome these biases different applications of linear models have been chosen. Using regularization parameters and differently engineered features it becomes possible to observe the change in performance of these models given the input data. Even though the data and seven independent variables remain the same ones across all models, the exact form of these features changes due to engineering. However, the nature of the features does not change and inherent biases of for example "Overall Quality" of a house remain: It is a subjective statement made by humans and therefore is based on human bias. However, given the task is to predict house prices, we can assume that the sales price of houses in Ames is equally based on biased decisions.

Feature engineering class	**Description**	**Implementation**
Missing values	Replacing missing values with mean (numerical features) and	The chosen features do not contain missing values however this technique is

	mode (categorical features) of the features values	important to minimize distortion and imbalances in the data set that can lead to false predictions
OneHotEncoding	Changing categorical features into numerical ones	In order to be able to use categorical inputs for ML models, such as street types, they need to be translated to numerical values
Normalization	Normalization is a technique to address the differences of numeric values in features and normalize it to a range of 0 to 1	The paper deploys normalization to address the different magnitude of the numerical features and achieve “equal contribution” by each feature to the model.
Scaling	Like normalization, feature scaling simply puts features with different numerical scales on the same scale	The paper uses scaling as a way to address different numerical scales of features however different to normalization without achieving “equal contribution” of the features.
Bucketizing	Bucketizing is the technique of reducing the space of a feature to more comprehensive buckets, e.g. bucketing square footage into 4 groups.	The models in this paper use bucketizing to account for the wide spread of square footages of some of the features.
(Vectorizing)	Vectorizing is a technique used to compile all feature values of an instance into a single vector	The model uses vectorization to create feature vectors with a label for every instance in order to use the Spark ML library for model training.

5 Machine learning experiment setup

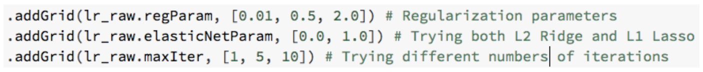
In the present case, the data set is largely described by the following characteristics:

1. 1460 instances
2. 7 features
3. 1 dependent variable

For modeling purposes, the data set is split into a training set and validation set using a 90% (training)-10% (validation) split. This step is repeated four times according to the different chosen feature engineering techniques. This defines the number of models:

1. “Raw features model”: OneHotEncoding applied to the categorical features, keeping raw numerical features
2. “Normalized Model”: Normalized the numerical features, onehotencoded the categorical features
3. “Bucketized model”: Bucketized the numerical features, onehotencoded the categorical features
4. “Scaled model”: Scaled the numerical features, onehotencoded the categorical features

The experiment uses a 5-fold cross-validation model to test advanced linear models across a range of hyper-parameters. Both Lasso and Ridge Regression are used and tested across a multitude of variations. Ultimately, the best model is chosen to extract the RMSE performance metric of the model. The hyper-parameter grid for the cross-validation models remains the same across all four models:



```
.addGrid(lr_raw.regParam, [0.01, 0.5, 2.0]) # Regularization parameters
.addGrid(lr_raw.elasticNetParam, [0.0, 1.0]) # Trying both L2 Ridge and L1 Lasso
.addGrid(lr_raw.maxIter, [1, 5, 10]) # Trying different numbers of iterations
```

There are multiple reasons for choosing this setup:

1. Good performance in generalizing across the data

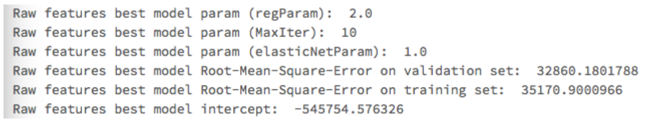
2. Cross validation is chosen to provide model results that are more reliable when predicting on data that hasn't been seen by the model
3. Lasso and Ridge regression are chosen to alleviate potential multicollinearity amongst the predictor variables in the models, especially given the inherent correlation of the features in the Ames dataset

6 Experiment outcomes and model performances

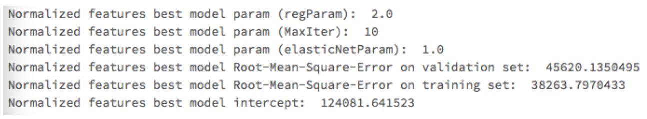
The root-mean-square-error has been indicated by the Kaggle competition to be chosen as evaluation metric. This makes sense given the good interpretability of the metric. It is used to measure the difference between observed sales prices and the sales prices predicted by the model. According to the experiment setup, the cross-validation method led to different models chosen to best predict sales price across the different feature engineering techniques:

1. "Raw Feature Model": 10 iterations, with a 2.0 regularization parameter and using the Lasso Regression
2. "Normalized Model": 10 iterations, 2.0 regularization parameter and using the Lasso Regression
3. "Bucketed Model": 10 iterations, 0.5 regularization parameter and using the Lasso Regression
4. "Scaled model": 1 iteration, 2.0 regularization parameter and using the Ridge Regression

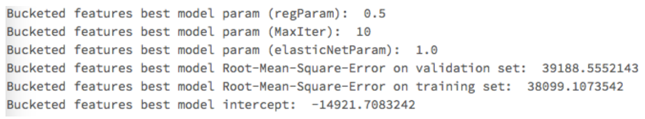
Firstly, it is interesting to see how three out of four models chose the Lasso Regression and 10 iterations and only the scaled model, which in theory is similar to the raw and normalized model, ended up performing best using a Ridge Regression and 1 iteration. This result is likely due to the fact that in the scaled model, the overall quality feature was highly penalized given its much lower numerical value (before scaling) as opposed to square footage and year built. In the normalized model, this effect is minimized due to the normalization factor. Overall, all models perform reasonably well, with the best model performance shown by "raw model". An explanation is that this model provides the most biased data set (taking over all inherent biases of the features) which generally favors linear models that generalize well across the problem space. Interestingly, the "bucketed model" is not the one performing the best. Given the small data set it would've been intuitive that adding features to the model to reduce bias would have increased its performance. However, feature reduction techniques could have had a bigger impact. Since bucketing is such a technique that reduces a feature to a bucket the assumption would have been a higher performance outcome. However, bucketing of a feature does not seem to affect model performance as actually eliminating a feature. Looking at the outcomes of the models based on data segments, it becomes obvious that certain techniques (onehotencoding) benefit most the categorical segments of the dataset whereas normalization or bucketing benefit most the numerical features of a data set. To take this exercise further it would be worthwhile trying additional model types such as Support-Vector-Machines or XGBoost. Additionally, one could work with the entire feature space of the data set and use selection techniques to use a more accurate selection of features. As to feature engineering techniques: Bucketing still seems the intuitive best choice for many of the numerical features and normalizing feature also is recommended.



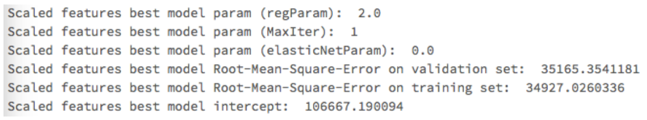
```
Raw features best model param (regParam):  2.0
Raw features best model param (MaxIter):  10
Raw features best model param (elasticNetParam):  1.0
Raw features best model Root-Mean-Square-Error on validation set:  32860.1801788
Raw features best model Root-Mean-Square-Error on training set:  35170.9000966
Raw features best model intercept:  -545754.576326
```



```
Normalized features best model param (regParam):  2.0
Normalized features best model param (MaxIter):  10
Normalized features best model param (elasticNetParam):  1.0
Normalized features best model Root-Mean-Square-Error on validation set:  45620.1350495
Normalized features best model Root-Mean-Square-Error on training set:  38263.7970433
Normalized features best model intercept:  124081.641523
```



```
Bucketed features best model param (regParam):  0.5
Bucketed features best model param (MaxIter):  10
Bucketed features best model param (elasticNetParam):  1.0
Bucketed features best model Root-Mean-Square-Error on validation set:  39188.5552143
Bucketed features best model Root-Mean-Square-Error on training set:  38099.1073542
Bucketed features best model intercept:  -14921.7083242
```



```
Scaled features best model param (regParam):  2.0
Scaled features best model param (MaxIter):  1
Scaled features best model param (elasticNetParam):  0.0
Scaled features best model Root-Mean-Square-Error on validation set:  35165.3541181
Scaled features best model Root-Mean-Square-Error on training set:  34927.0260336
Scaled features best model intercept:  106667.190094
```

ASSOCIATED CONTENT

In addition to this document there is a working PySpark code example conducting the above analysis. To publicize the code it has been uploaded to a github repository whose link is below.

REFERENCES

[1] Gordon, Vitaly. The Lego Model for Machine Learning. In *YouTube*, 2016

[2] Marcelino, Pedro. Private release on Kaggle Competition Kernels in *Comprehensive data exploration with Python,* 2013

[3] Papiu, Alexandru. Private release on Kaggle Competiton Kernels in *Regularized Linear Models*, 2013

[4] Dean De Cock. Kaggle Competition Description, in *House Prices: Advanced Regression Techniques*, 2013

[5] Farcaster. What is Data Science and What Does a Data Scientist Do?. In *KDnuggets,* 2017