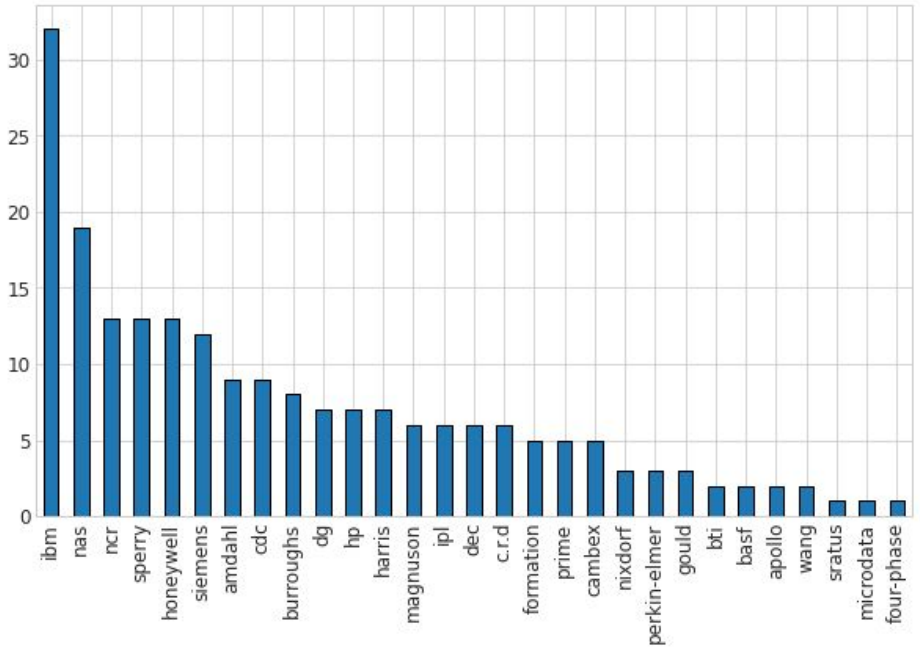
Final Course Project: Estimating CPU Performance

29. January 2021
by Tom K. Walter

1. Dataset and Goal

For my project, I am using the Computer Hardware Dataset, which was originally collected and analyzed by Phillip Ein-Dor and Jacob Feldmesser in their paper *"Attributes of the performance of central processing units: A relative performance prediction model"* in the Communications of the ACM (1987).[1] It is available on Kaggle and the UCI Machine Learning repository.[2] The dataset contains 208 instances, which correspond to 208 different CPU models from 29 vendors (i.e. manufacturers, see Image 1); and 10 features. These features include 6 numerical features, 2 non-predictive variables, 1 target as well as the predictions from Ein-Dor and Feldmesser's stepwise linear regression model. The numerical features include a CPU's main memory size, cache memory size, number of channels, and machine cycle time (further explained in Table 1). The target variable PRP is the CPU's *published relative performance*, which is a comparable standard used by computer manufacturers at the time.

Image 1: Number of CPU-models by Manufacturer



The associated task of the dataset is to predict the relative performance of the CPUs. Moreover, the dataset is unique because it comes with authors' *estimated relative performance* (ERP) alongside the usual target variable. This provides an opportunity to use the predictions from Ein-Dor and Feldmesser as a benchmark to compare other regression models against. Therefore, the *goal of my project* focuses not on the interpretability but on the *accuracy* of further models.

[1] Ein-Dor, Phillip, and Jacob Feldmesser. "Attributes of the performance of central processing units: A relative performance prediction model." *Communications of the ACM* 30, no. 4 (1987): 308-317.
[2] For Kaggle, see https://www.kaggle.com/faizunnabi/comp-hardware-performance. For the UCI repository, see https://archive.ics.uci.edu/ml/datasets/Computer+Hardware.

Table 1: Dataset Description

Variable	Description
VENDOR	name of the manufacturer
MODEL	unique name of CPU model
MYCT	machine cycle time in nanoseconds (integer)
MMIN	minimum main memory in kilobytes (integer)
MMAX	maximum main memory in kilobytes (integer)
CACHE	cache memory in kilobytes (integer)
CHMIN	minimum channels in units (integer)
CHMAX	maximum channels in units (integer)
PRP	published relative performance (integer)
ERP	estimated relative performance from the original article (integer)

The utility of developing an accurate model that estimates the performance of a CPU from its characteristics is either, for users, to compare CPU-models without needing to benchmark their own machines or, for manufacturers, to estimate the performance of a CPU-model that is still in production.

The next steps are feature engineering, exploratory data analysis, the creation of a set of benchmark metrics as well as the development of various regression models. For comparison, I will develop a linear, lasso, ridge, and elastic-net model. To optimize the model accuracy further, I will utilize Sklearn's k-fold and grid-search cross validation functions.

2. EDA and Feature Engineering

2.1. Data Cleaning

Before feature engineering the dataset has been checked for missing values and duplicate entries. None have been found.

2.2. Feature Engineering

Instead of just taking the manufacturer's specifications about the CPUs for granted, I will transform the variables into metrics that more closely resemble the actual functioning of a CPU. Thus, feature engineering can help to improve both the interpretability and the accuracy for regression analysis.

The features CHMIN and CHMAX are both referring to the same underlying feature of channels and express a range; they will be transformed to express the average number of channels per CPU (CH_AVG). Similarly, the MMIN and MMAX feature are transformed to instead express average memory, and from kilobyte to megabyte, as MEM_AVG_MB. CACHE is also converted from kilobyte to megabyte as CACH_MB. MYCT, which is the machine cycle time in nanosecond inverted to show machine cycles per nanosecond, i.e. the SPEED. The SPEED and CH_AVG are multiplied by each other to provide the number of channel executions per nanosecond (CH_CAP).

A mathematical summary and description of this feature engineering process are provided in Table 2 and the resulting summary statistics are in Table 3. Before the features are put into regression, they are also standardized (setting their mean to 0, and their standard deviation to 1) and polynomial features of the second and third degree will be created.

Table 2: Feature Engineering

Feature	**Transformation**	**Description**
CACH_MB	= CACHE/1000	cache memory in megabytes
CH_AVG	= ((CHMIN+CHMAX)/2)+1	average number of channels
SPEED	= 1/MYCT	machine cycles per nanosecond
CH_CAP	= CH_AVG*SPEED*10	channel executions per 10 nanosecond
MEM_AVG_MB	= ((MMIN+MMAX)/2)/1000	average memory in megabytes

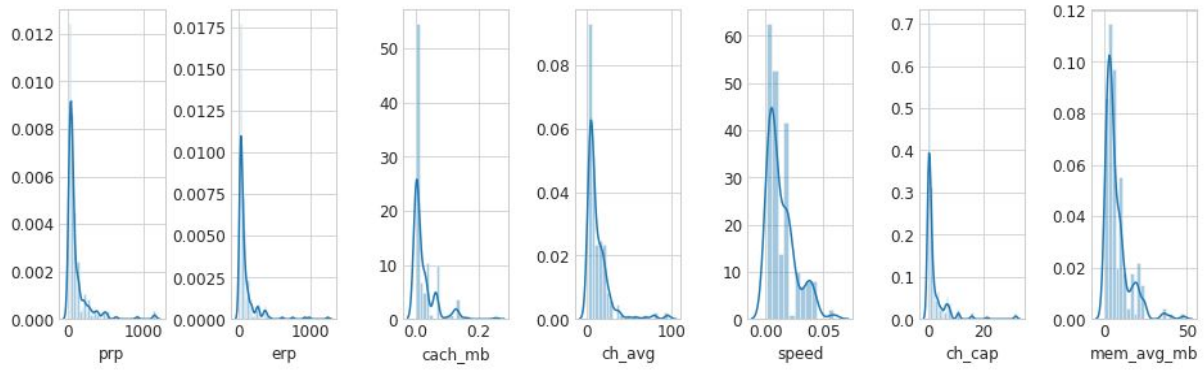
Table 3: Summary Statistics

	PRP	ERP	CACH_MB	CH_AVG	SPEED	CH_CAP	MEM_AVG_MB
Count	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000	208.0000
Mean	105.1779	98.8510	0.0241	12.1923	0.0136	2.2653	7.3523
Std	161.0902	154.9750	0.0374	14.5770	0.0122	4.2206	7.4541
Min	6.0000	15.0000	0.0000	1.0000	0.0007	0.0067	0.0640
25%	27.0000	28.0000	0.0000	4.0000	0.0044	0.2255	2.5000
50%	49.5000	45.0000	0.0080	6.5000	0.0091	0.8214	5.0000
75%	111.5000	99.5000	0.0320	16.2500	0.0200	2.1750	9.0000
Max	1150.0000	1238.0000	0.2560	95.0000	0.0588	31.6667	48.0000
Range	1144.0000	1223.0000	0.2560	94.0000	0.0582	31.6600	47.9360

2.3. Exploratory Data Analysis

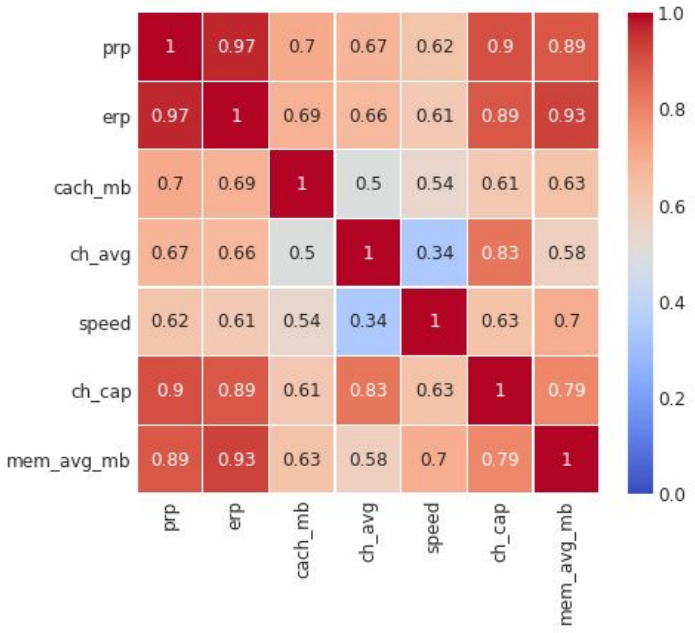
After having done feature engineering, it makes sense to explore the data further to identify other potential flaws and determine if the features can be considered good predictors for regression analysis. Ideally, for regression analysis, all features and the target should be normally distributed or, at least, they should all have similar distributions. Although the attributes and target are not normally distributed, they all possess a similar right skew as shown in Image 2. That means there are a few high-performance CPUs as outliers in each category. Dropping these instances would reduce the already small number of observations, so they are kept. In order to prevent the skew from affecting the modelling, the data will be standardized before modelling.

Image 2: Distribution of the Variables



A better way to estimate whether or not the features are good predictors of the target is to determine the correlation between each feature and the target. A high absolute correlation coefficient between a feature and the target means it is generally a good predictor. In Image 3, the correlation matrix illustrates that all features have a strongly significant, positive correlation with the target.

Image 3: Correlation Matrix



3. Benchmark Metrics

As mentioned before, the dataset includes the target variable PRP and the predictions ERP obtained by Ein-Dor and Feldmesser from their stepwise linear regression model. Using these as y and y_{pred}, common accuracy metrics for regression such as R-squared, mean squared error (MSE), root mean squared error (RMSE), and mean absolute error (MAE) can be computed for their model as displayed in Table 4. Now, these metrics may serve as a benchmark for comparison, when it comes to improving the accuracy of regression models from the given dataset. It is worth mentioning that their original model achieved relatively high accuracy scores. In Image 5, the left scatter plot of PRP vs ERP also confirms this. However, with regards to machine learning, doubts can be raised whether their model overfits the data.

4. Modelling and Evaluation

In order to model the data and improve accuracy scores without overfitting, I will try several regression techniques: ordinary linear, lasso, ridge, and elastic net. I will further utilize Sklearn's k-fold and grid-search cross validation functions towards this end. Regardless of technique, each model will be given standardized data. The grid search will test:

- for the best degree of polynomial feature, in range of [1, 2, 3];
- for the best penalty term in all regularized models, in range of [0.1, 1, 10, 100];
- for the best l1-l2 ratio in the elastic net, in range of [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9];
- and all models will be optimized on a k-fold split, where k=4.

Table 4 displays the accuracy scores for each model and their optimized hyperparameter in comparison to the benchmarks derived from Ein-Dor and Feldmesser's model. Even with k-fold cross validation, the linear regression performed similarly to the benchmark in terms of R-squared, but the

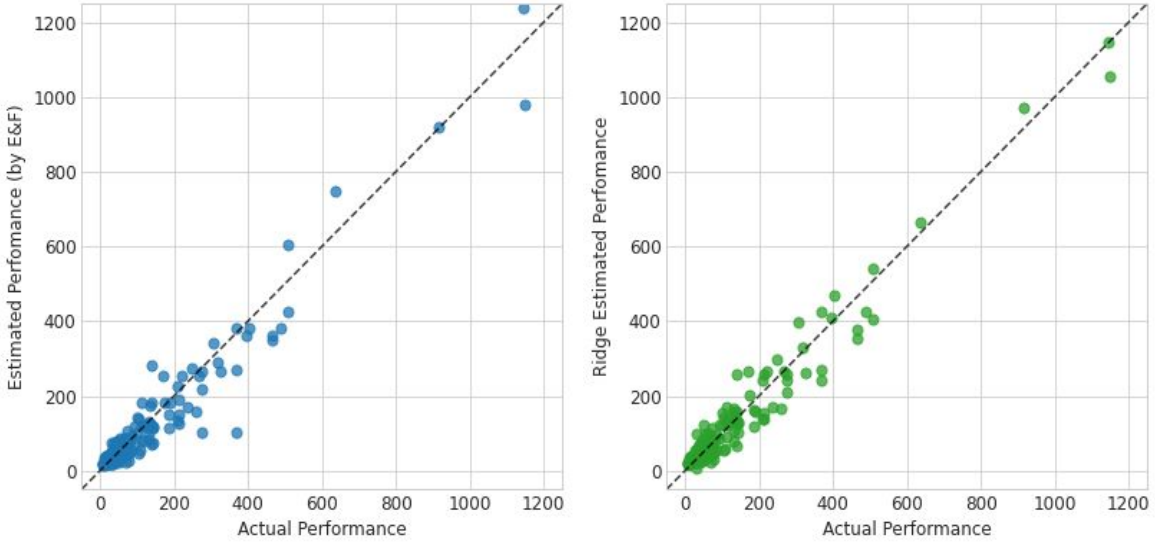
MSE, RMSE, and MAE are slightly worse. This means that Ein-Dor and Feldmesser's model was indeed very accurate and did not overfit the data.

Although the regularized models performed better on all scores than the benchmark, it is on average a small improvement of 0.02 in terms of R-squared. But in terms of MSE and RMSE, the regularized and cross-validated models perform better than the benchmark by a greater degree. Among each other, the regularized models all perform nearly identical, but the best performing model is the ridge regression. To visualize the improvements in accuracy over the Ein-Dor and Feldmesser model, its predictions are also visualized on the right scatter plot in Image 5. The comparison shows that a penalty applied to the coefficients has led to a better bias-variance trade-off, especially the high-performance outlier CPU-models can now be better predicted.

Table 4: Model Evaluation

	Benchmark Metrics	**Linear Regression**	**Lasso Regression**	**Ridge Regression**	**Elastic Net Regression**
R^2	0.932404	0.931615	0.955496	0.955888	0.953945
MSE	1745.682692	1766.072135	1149.336872	1139.204192	1189.386036
RMSE	41.781368	42.024661	33.901871	33.752099	34.487477
MAE	24.442308	28.229341	23.057861	22.505444	23.064060
Best Hyper-Params	N/A	poly=1	poly=2 alpha=1	poly=2 alpha=10	poly=2 alpha=0.1 ratio=0.1

Image 5: Actual vs Prediction from E&F's Model (left) vs from Ridge Regression (right)



5. Conclusion

In order to utilize such CPU performance prediction models as intended, it was important to build them with cross-validation to prevent them from overfitting on the training data and then mis-estimating on new, unseen data. Both regularization and cross-validation have thus led to higher and more robust accuracy scores than the original model by Ein-Dor and Feldmesser. For future analysis of this dataset, it could be useful to convert the VENDOR variable into dummy variables in order to capture the difference in CPU performance by manufacturer. In turn, this could help customers to pick a CPU-manufacturer that is in their preferred performance-and-price range. Alternatively, models could be built with the goal of interpretability in order to provide customers with better insight into what features impact CPU performance and in what way.