COMP 309/AIML421 | Machine Learning Tools and Techniques

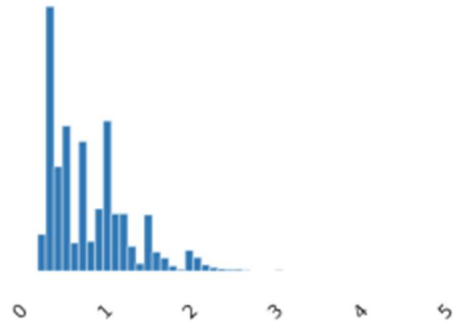
Assignment 4, Corvin Idler - Student ID 300598312

Performance Metrics and Optimisation (Part A)

Part 1: Performance Metrics in Regression [30 marks]

According to the pandas-profiling tool there were some missing values in the dimensions of the diamonds:
8 zeros on x, 7 zeros on y, 20 zeros on z
I decided to fill the values with a MICE type imputation.

As quite some features were highly skewed e.g. carat (see below)

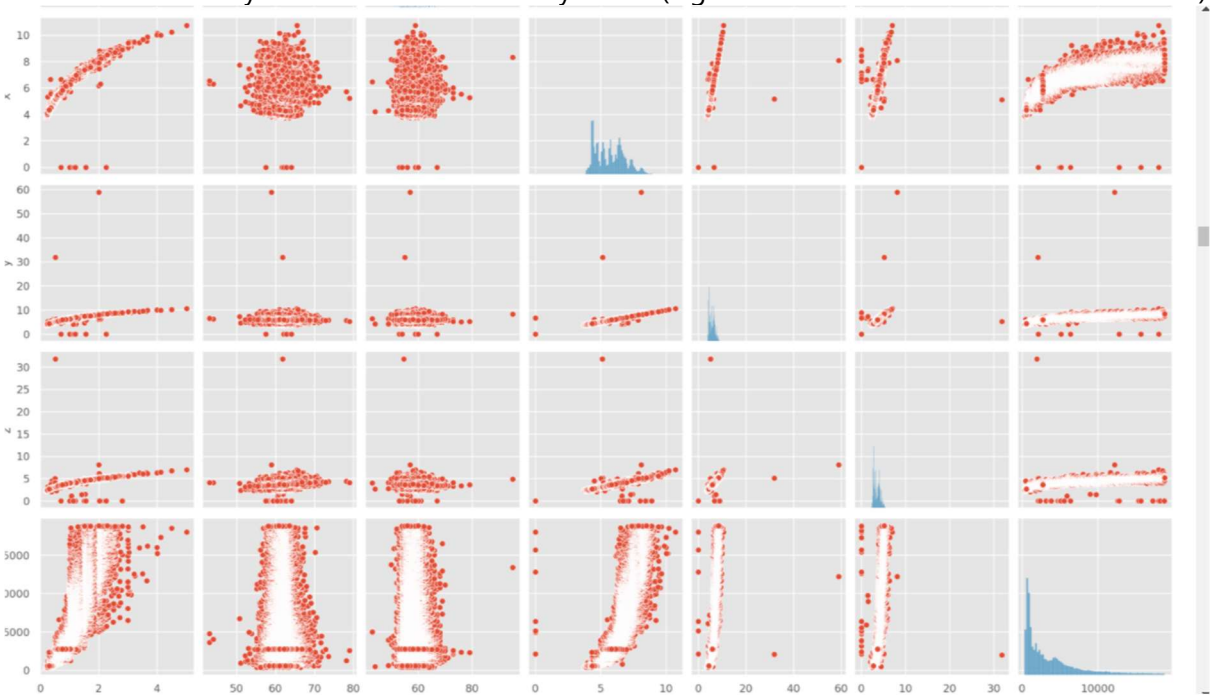


I felt like I needed to normalize them beyond plain vanialla standardisation. I tried box-cox transformation and quantile transformation. The quantile transformation achieved the best normalisation (c.f. examples below)

That said, I found that box-cox performed best in a linear regression in terms of regression performance. I can not know if that results would hold true for all algorithms but due to a lack of computing power I had to settle for one data set and decided for box-cox transformed (incl. output variable). For the calculation of performance metrics I inversed the box-cox transformation on the predictions.

Further steps I took was to one-hot encode categorical variable and split the data set 70%/30% into training and test data set.

I did an outlier analysis and there are certainly some. (e.g. c.f. some variables in screenshot below)



Many of them are due to the missing entries (zero on x, y, z) and a lot of the others are univariate outliers (in one dimension). Given the amount of overall datapoints I don't think I need to spent too much time eliminating the outliers from the training set and I would be tempted to do this (if at all) only for multivariate outliers... which complicates a bit the outlier identification (e.g. statsitical multivariate quality control algorithms or machine learning algorithms like single forest etc). Long story short, I decided to leave things and see what happens.

Report of results:

	Algorithm	RMSE	MSE	MAE	RSE	R2
0	Linear regression (OLS)	1,148.23	1,318,428.47	738.57	0.08	0.96
1	Linear regression (OLS) with Box-Cox	823.67	678,436.89	410.11	0.04	0.98
2	Linear regression (OLS) with quantile transfor...	908.93	826,146.95	531.53	0.05	0.97
3	Linear regression (Ridge)	825.08	680,758.18	410.50	0.04	0.98
4	Decision Tree	661.56	437,655.02	325.74	0.03	0.99
5	KNN	759.88	577,417.47	359.85	0.04	0.98
6	RF	555.55	308,636.73	272.15	0.02	0.99
7	Gradient Boosted Decision Trees	547.96	300,256.19	265.74	0.02	0.99
8	linear regression SGD	823.76	678,580.22	410.25	0.04	0.98
9	linear support vector regression	838.18	702,539.93	412.60	0.04	0.98
10	polynomial support vector regression	1,581.64	2,501,588.53	937.68	0.15	0.92
11	radial basis function support vector regression	826.99	683,915.03	477.55	0.04	0.98
12	MLP	554.84	307,846.51	279.51	0.02	0.99

Performance of different algorithms:

Obviously ensemble based algorithms (Random Forrest and Gradient Boosted Decision trees perform really well. It's a slightly unfair comparison to pitch ensembles of classifier against singular classifiers). That said, the MLP stacks up really well which is also not too surprising given it's capability to approximate non-linear relationships.

I was surprised how well a singular decision tree stacks up, given it's relative simplicity and interpretability (e.g. compared to a linear support vector machine). I was really negatively surprised by the underperformance of the polynomial support vector regression as this is quite a powerful algorithm (maybe a case of overfitting). The RBF SVM doesn't stack up much better than the linear SVM, so a good learning experience that complexity doesn't necessarily lead to better results. KNN came in in the upper middle range of performance and was at least better than linear regression.

I was impressed by how much difference the normalisation of features made (at least for the linear regression) and also that the quantile transformation which leads to optically better normalisation of the features didn't lead to better regression results. Again, lesson learnt, that complexity or "sophistication" doesn't always lead to better performance.

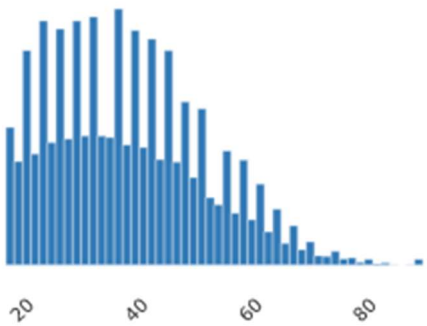
Quick discussion of the metrics:

Obviously the RMSE and MSE are related by the square root, so they have different order of magnitude but the order of the results is the same. R2 I threw in only for completeness.

The ranking between RMSE and MAE can be different. E.g. on RMSE the MLP comes in second, but ranked by MAE the Random Forest comes in second. Therefore these two measures are not redundant. (unlike RMSE and MSE). The RSE is rounded too harshly to make a similar assessment.

Part 2: Performance Metrics in Classification [30 marks]

Pre-processing needs: a lot of the numerical features were non-normal and therefore I decided to not only standardize them but also perform a box-cox transformation to bring them closer to a normal distribution. Particularly distance based algorithms like KNN might benefit from this. Below the distribution of age.



A second important pre-processing step was the removal of redundant features (e.g. education and education-num). I dropped education-num. Below frequency table proves the redundancy.

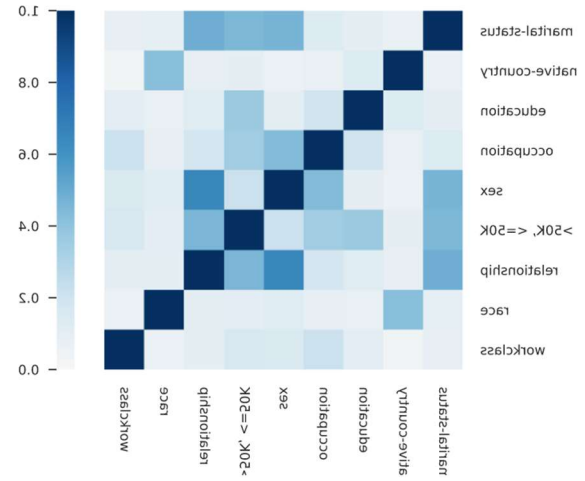
education education-num

10th	6	933
11th	7	1175
12th	8	433
1st-4th	2	168
5th-6th	3	333
7th-8th	4	646
9th	5	514
Assoc-acdm	12	1067
Assoc-voc	11	1382
Bachelors	13	5355
Doctorate	16	413
HS-grad	9	10501
Masters	14	1723
Preschool	1	51
Prof-school	15	576
Some-college	10	7291

As per https://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf fnlwgt could be treated in various ways. It should certainly not be a predictive variable for income as it just indicates how many people in the census are estimated to share this feature combination one could and maybe should use it for weighted classification (sample weighting) or "bootstrapping" to make sure the data set and estimates are representative of the whole population. For time reasons I decided to not treat if further apart from excluding it as a predictive feature.

I "merged" the two capital features to calculate the net change and removed the two original features. As per the above publication relationship is also highly redundant and I therefore excluded it as well to save computing time.

I also dropped relationship as per the findings (high correlation with marital status) on page 2 of this pape r https://cseweb.ucsd.edu/classes/sp15/cse190-c/reports/sp15/048.pdf and the correlogram below.



I did some basic level merging of categorial variables to reduce the number of features to a manageable level once I one-hot encode them. Namely: native country levels were collapsed to US and non-US (an alternative solutions could have been to distinguish between the developed world and developing countries). Education levels were collapse into non-High School, High School, Graduate Studies, Associate Degrees (undergraduate). Occupation was collapsed to white-collar and blue-collar workers. This made computations finally achievable.

Lastly, I one hot encoded all categorical variable to make sure they can be processed by all algorithms I wanted to evaluate.

Results:

	Algorithm	accuracy	precision	recall	F1-score	AUC
0	KNN	0.792273	0.764691	0.542820	0.634931	0.729755
1	Naive Bayes	0.483938	0.955018	0.308604	0.466472	0.634548
2	Bagged SVC	0.823844	0.726209	0.606120	0.660752	0.757956
3	simple RBF SVC	0.799275	0.809152	0.551187	0.655710	0.741085
4	linear SVC	0.803083	0.734789	0.563847	0.638067	0.736666
5	simple Decision Tree	0.814692	0.718929	0.588173	0.647011	0.747411
6	Random Forrest	0.823844	0.752470	0.601663	0.668669	0.759336
7	Adaboost on Decision Tree	0.826423	0.866355	0.590361	0.702213	0.771020
8	Gradient boosted decision trees	0.854984	0.733229	0.678700	0.704912	0.797044
9	LDA	0.768012	0.853094	0.505313	0.634684	0.723795
10	MLP	0.811314	0.835413	0.568471	0.676563	0.754459
11	Logistic Regression	0.786008	0.832553	0.529957	0.647654	0.733530

Discussion:

The best two algorithms in terms of performance metrics Gradient Boosted decision trees and AdaBoost with simple decision trees. The two algorithms are similar in that they are both ensemble methods based on decision trees, but adaboost optimized an exponential cost function and is therefore more prone to outliers. Gradient Boosting is more generic in terms of which cost function it can optimize (as long as its differentiable).

Part B: Optimisation

See attached Assignement Part B.ipynb file