A Tour of Machine Learning Techniques: A Comparison Between Tree Based and Neural Network Methods for Classification and Regression

Marc Ishak
Student
UNSW
Sydney, Australia
m.ishak@student.unsw.edu.au

***Abstract*—A variety of Neural Net and Tree models are explored on their classification and regression ability.**

***Index Terms*—Neural Net, Tensorflow, Keras, Random Forest, Pruning, Trees, Sci-Kit Learn**

I. Introduction

As interest in neural networks grows in part due to the rising popularity of deep learning in its applications in facial recognition [1], natural language processing [2] and game playing [3]. This paper aims to find whether these neural net techniques out-preform more traditional tree based methods [4], namely CART [5] and random forest [6], in classification and regression across the 'Parkinson Speech Dataset with Multiple Types of Sound Recordings' [7] and 'The Population Biology of Abalone (Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and the Islands of Bass Strait' [8] datasets respectively.

This paper acts as an extension of the engineering process discussed in 'Grid Search Across Neural Net Machine Learning Models for Classification', extending the scope from comparing Keras based neural net models to each other for classification to comparing keras models to scikit-learn models across both classification and regression.

II. Methodology

A. Reproducability

A `conda` [9] environment was created to handle package management and compatibility. An `environment.yml` file was created to help ensure that the results and methodology explored is reproducible.

Data cleaning, data perpetration, model building and fitting, and assessment for both data sets was developed to work as a single directed acyclic graph made up of two pipelines (data engineering and data science) composed with pure functions. The goal of this is to minimise issues of state in regards to the reproducibility of the model that can often occur with more ad-hoc styles of development structure. The data engineering pipeline consists of data cleaning and data preparation sub-pipelines, the first of which aims to reduce any unwanted imperfections (like extra columns) that pre-exist in the dataset and the second which aims to augment the data into something more ideal for the algorithms at hand. After going through these pipelines, the data is then saved into csv files for any other analysis. After passing through the data engineering pipeline, the data is then subject to the data science pipeline which fits machine learning models and then assess them on their prediction accuracy. Model information and assessment results are then written to a csv for analysis. The .csv files are then analysed in jupyter notebooks [19] using pandas [18], statsmodels [26] and matplotlib. [20]

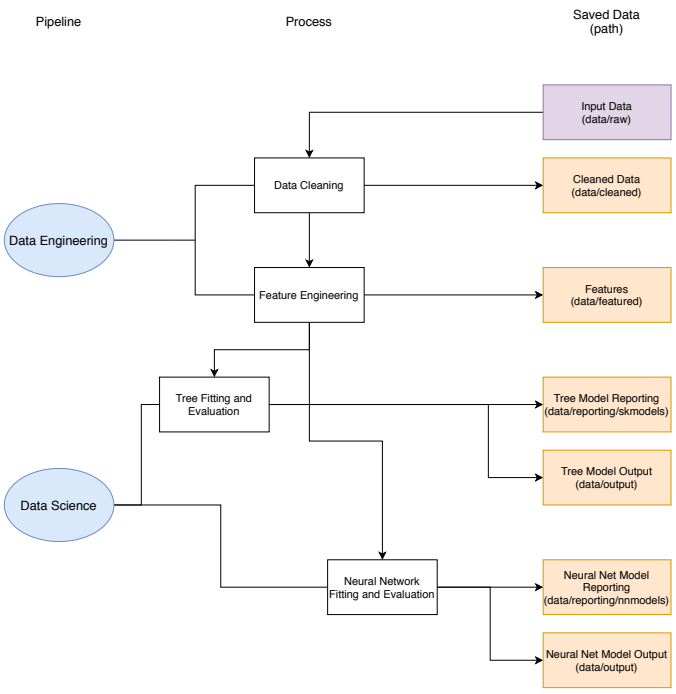

Fig. 1. Data pipeline Structure

B. Data Preparation - Parkinson's Data

Note: This section has been reused from a previous paper "Grid Search Across Neural Net Machine Learning

Models for Classification"

After initial data cleaning (as guided [10]), exploratory data analysis (EDA) [11] utilising the python package `pandas-profiling` [12]. Utilising `pandas-profiling`, it became clear that a majority of the variable in the dataset were highly correlated.

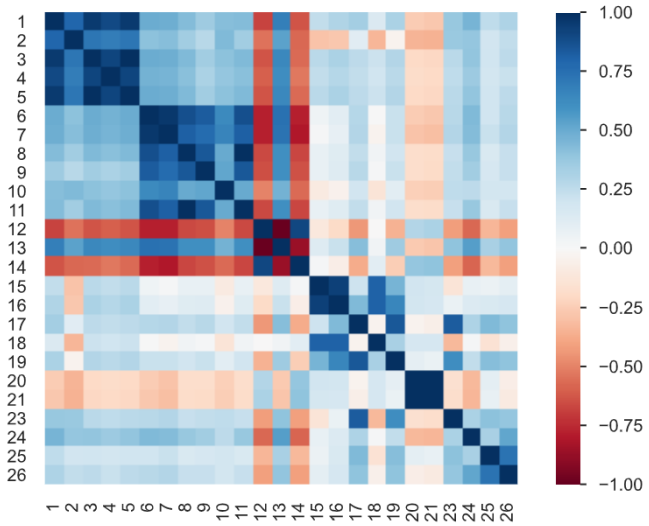

Fig. 2. Correlation Plot for variables in Parkinson's dataset

As such `scikit-learn` [14] was utilised to carry out min-max scaling and principle component analysis (PCA) [15] was used to reduce the number of input variables down from twenty six to six (as per the elbow method from the scree plot). Approximately 87.86% of the variability of the data was accounted for in theses 6 variables. Fundamentally, reducing the number of input variables helps reduce the complexity of the system, increasing its computational and (minimally) memory efficiency.

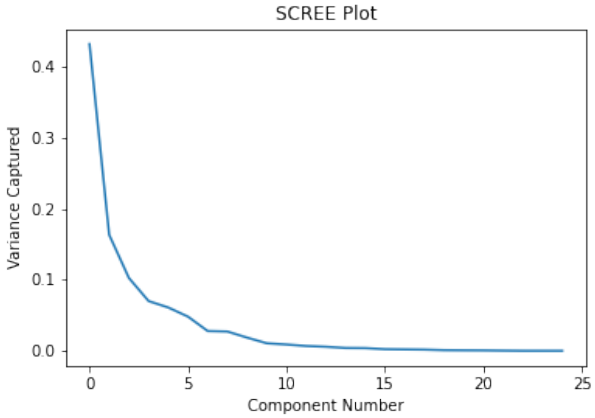

Fig. 3. Scree Plot showing proportion of variance explained per component for the Parkinson's dataset

After the PCA transformation, a train-test-split was carried out (with a fixed seed) with a test size of 20% of the size of the data. Once completed, the split data was written to disk in the comma separated values (.csv) format. Furthermore, the training data was then split into training and validation sets at a 70/30 ratio for the model fitting process.

C. Data Preparation - Abalone Data

The abalone dataset had similar properties Parkinson's dataset, namely a large number of variables with a high correlation with each other. Hence it followed a similar process, with min max-scaling followed by PCA on the numerical variables. In this case we found that a single component could account for 97.41% of the variance across those features.

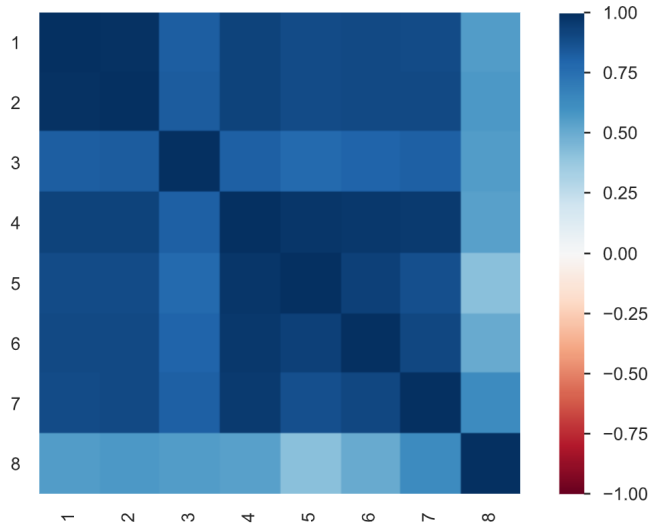

Fig. 4. Correlation Plot for numerical variables in abalone dataset

However, the abalone data set also consists of a single nominal categorical variable - Sex. This was encoded using one hot encoding [13] as the data is not ordinal and other encoding methods to imply extra (false) information to the algorithm.

D. Wrapper Libraries

Note: For more information on the wrapper libraries (particularly for Neural Net Models) see "Grid Search Across Neural Net Machine Learning Models for Classification"

1) The `ModelWrapper` Class: The `ModelWrapper` class only exists for neural net models, and attempts to provide a `scikit-learn` style interface for the process of building neural net models in Keras [17]. It contains functions to build, fit and compile the model along with descriptive model plotting abilities. By having this wrapper, model management becomes easier as the wider state of the model is constrained to its own object, helping reduce the complexity of errors.

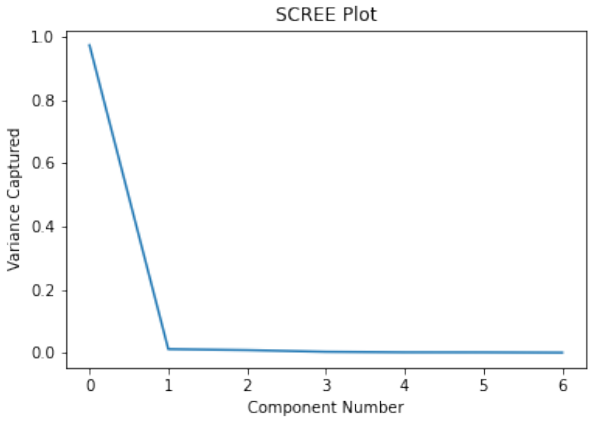

Fig. 5. Scree Plot showing proportion of variance explained per component for the abalone dataset

2) The `ModelInstance` *Classes:* The `ModelInstance` classes are for both neural net (in conjunction with the `ModelWrapper` class) and tree based models. They handle the data as well as the model to encapsulate both of them together. It also stores details about the models performance such as the false positive rate, the true positive rate and f1 score for classification models but also the R^2 score and RMSE for regression model and handles key plots determining the accuracy and effectiveness of that model on that dataset.

E. Tree Model Fitting and Evaluation

Tree model fitting was extended beyond the standard `scikit-learn` .fit() method to incorporate cost complexity pruning [21] directly into the fitting process. The training data is first split 70/30 into training and validation sets, then an initial naive model is fitted to the dataset after which the range of alphas are extracted and sampled (to reduce the amount of models made, this parameter is tunable and is set to 100 by default). After which, new models are fitted on the training data with the alpha parameters, after which a plot of the training and validation accuracy is constructed and the model with the highest validation accuracy is selected.

Tree models revolved around 2 distinct approaches: Decision Trees and Random Forests

F. Neural Network Model Fitting and Evaluation

Similarly to the tree model fitting process, the training data was split 70/30. However, the built-in `batch_size` parameter was used to fit the data in ”batches” [24] at a fixed size of 300. This is a regularisation technique to reduce over fitting (and helps balance out the over fitting of cost complexity pruning in tree based methods).

Considering training time and accuracy (from ”Grid Search Across Neural Net Machine Learning Models for Classification”) 4 models were considered across 2 parameters:

- Optimisation Function: Adam, Stochastic Gradient Descent (SGD)

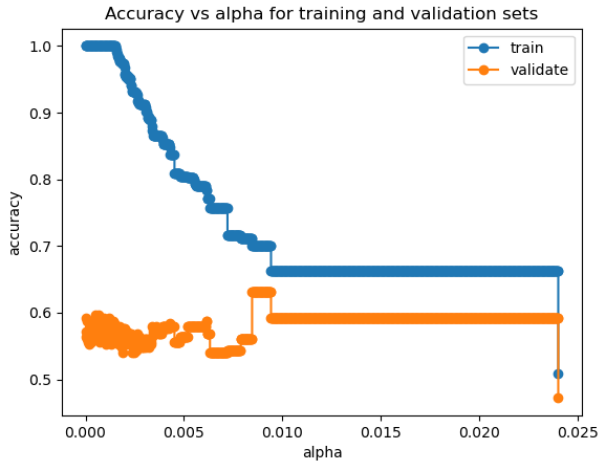

Fig. 6. Sample plot showing training and validation accuracy over sample alphas on a single Decision Tree Classifier

- Number of Hidden Layers: 2, 5

Otherwise, the number of Neurons per layer stayed fixed at 5 and a sigmoid activation was present on the output layer for classification problems.

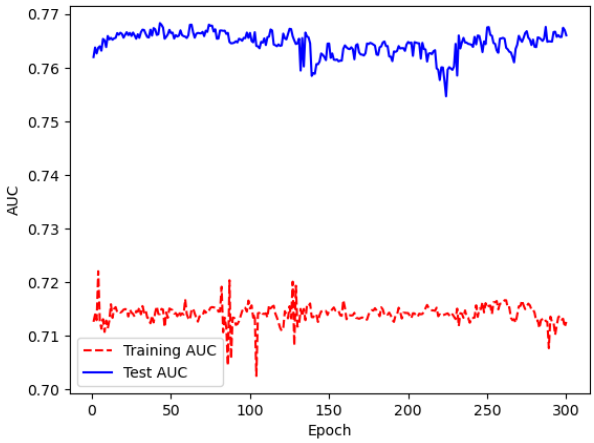

Fig. 7. Sample plot showing AUC of classifier neural net during the training process

III. Results and Discussion

A. Classification

	Optimisation Func	Neurons Per HL	Hidden Layers	Mean ROC-AUC Score	ROC-AUC Score Std. dev	ROC-AUC Score Upper 95% CI	ROC-AUC Score Lower 95% CI
0	Adam	5	2	0.663651	0.002951	0.669139	0.658164
1	Adam	5	5	0.500000	0.000000	0.500000	0.500000
2	SGD	5	2	0.642027	0.022453	0.683777	0.600277
3	SGD	5	5	0.500000	0.000000	0.500000	0.500000

	Optimisation Func	Neurons Per HL	Hidden Layers	Mean F1 Score	F1 Score Std. dev	F1 Score Upper 95% CI	F1 Score Lower 95% CI
0	Adam	5	2	0.646552	0.011667	0.669139	0.658164
1	Adam	5	5	0.135032	0.284672	0.500000	0.500000
2	SGD	5	2	0.622865	0.026554	0.683777	0.600277
3	SGD	5	5	0.270064	0.348651	0.500000	0.500000

	Optimisation Function	Neurons Per Hidden Layer	Hidden Layers	Mean r2 Score	r2 Score Std. dev	r2 Score Upper 95% CI	r2 Score Lower 95% CI
0	Adam	5	2	0.341932	0.000847	0.343507	0.340356
1	Adam	5	5	0.342675	0.002471	0.347269	0.338081
2	SGD	5	2	-0.000181	0.000005	-0.000172	-0.000191
3	SGD	5	5	-0.000779	0.001186	0.001426	-0.002984

	Optimisation Function	Neurons Hidden Layer	Hidden Layers	Mean MSE	MSE Std. dev	MSE Upper 95% CI	MSE Lower 95% CI
0	Adam	5	2	6.777991	0.008728	6.794219	6.761762
1	Adam	5	5	6.770329	0.025448	6.817648	6.723011
2	SGD	5	2	10.301694	0.000052	10.301792	10.301597
3	SGD	5	5	10.307847	0.012214	10.330559	10.285136

In classification, the best model was the Neural Net with 2 hidden layers and Adam optimisation, followed by the random forest model. Regression analysis shows that while Adam and Stochastic gradient descent neural net models often preformed comparably the choice of optimisation had a significant effect (at $\alpha = 0.05$) effect on the performance on the data in favour of Adam. The worst preforming models were neural nets with 5 hidden layers. This is unsurprising as increasing the layers makes finding the appropriate topological manifold distortions.

B. Regression

In regression, the best models were the neural nets with Adam optimisation. The number of hidden layers did not have a significant (at $\alpha = 0.05$) effect on the results on the dataset. Decision trees followed closely by. However, there is a large gap in performance between these methods and SGD Neural Nets and Random Forests. Stochastic gradient descent completely failed at predicting anything close and Random Forests only delivered about 30% of performance of decision trees. This is quite surprising, as random forests are by definition an amalgamation of decision trees. Identifying causal factors for this behaviour on both models would require more time and resources as they are both black-box models [27]. This could be a could potential vector for more research

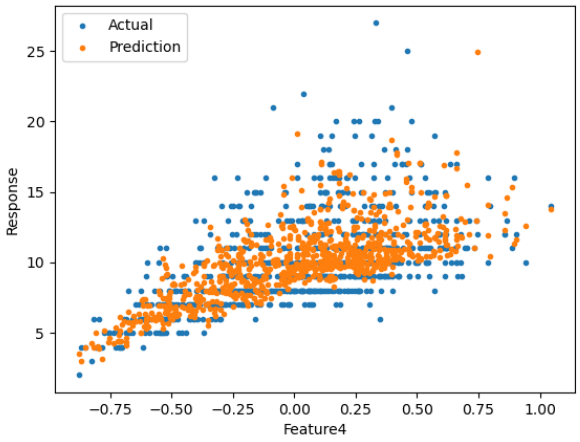

Fig. 8. Sample plot showing Prediction vs Actual on the single PCA variable from a random forest regressor

IV. Conclusion

This paper has taken a tour of neural net and tree methods and their comparative accuracy in classification and regression. Ultimately however, accuracy maximisation should not be the only metric in selecting a particular model, training time, interpertability, reproducability and scalability should all be considered for the variety of use cases and applications of machine learning today.

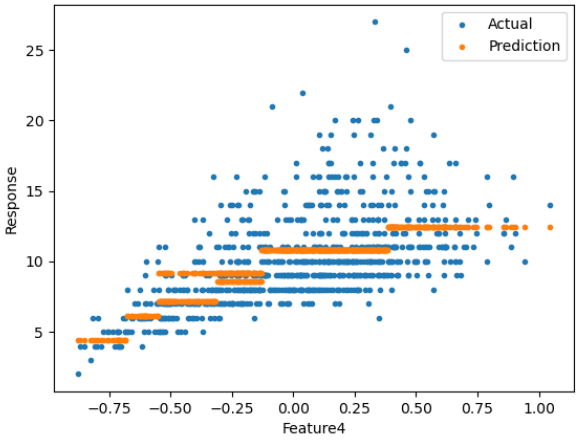

Fig. 9. Sample plot showing Prediction vs Actual on the single PCA variable from a decision tree regressor

References

[1] Sun, Y., Liang, D., Wang, X. and Tang, X., 2015. Deepid3: Face recognition with very deep neural networks. arXiv preprint arXiv:1502.00873.

[2] Young, T., Hazarika, D., Poria, S. and Cambria, E., 2018. Recent trends in deep learning based natural language processing. ieee Computational intelligenCe magazine, 13(3), pp.55-75.

[3] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C. and Józefowicz, R., 2019. Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

[4] Breiman, L., 1987. Classification And Regression Trees. BELMONT, CALIF.: WADSWORTH INTERNATIONAL Group.

[5] Steinberg, D. and Colla, P., 2009. CART: classification and regression trees. The top ten algorithms in data mining, 9, p.179.

[6] Liaw, A. and Wiener, M., 2002. Classification and regression by randomForest. R news, 2(3), pp.18-22.

[7] Erdogdu Sakar, B., Isenkul, M., Sakar, C.O., Sertbas, A., Gurgen, F., Delil, S., Apaydin, H., Kursun, O., 'Collection and Analysis of a Parkinson Speech Dataset with Multiple Types of Sound Recordings', IEEE Journal of Biomedical and Health Informatics, vol. 17(4), pp. 828-834, 2013.

[8] Nash, W.J., Sellers, T.L., Talbot, S.R., Cawthorn, A.J. and Ford, W.B., 1994. The population biology of abalone (haliotis species) in Tasmania. I. Blacklip Abalone (h. rubra) from the north coast and islands of Bass Strait. Sea Fisheries Division, Technical Report, 48, p.p411.

[9] Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda, Nov. 2016. Web. ¡https://anaconda.com¿.

[10] Chandra, R., 2020. Ed — Digital Learning Platform. [online] Edstem.org. Available at: ¡https://edstem.org/courses/4751/lessons/4575/slides/33747¿ [Accessed 20 November 2020].

[11] Tukey, J.W., 1977. Exploratory data analysis (Vol. 2, pp. 131-160).

[12] Martin Sotir, Joseph Yuen, Brian Lee, Stephanie Rivera, nscsekhar, abdulAziz, Pandas-profiling.github.io. 2020. Introduction — Pandas-Profiling 2.9.0 Documentation. [online] Available at: ¡https://pandas-profiling.github.io/pandas-profiling/docs/master/rtd/¿ [Accessed 29 October 2020].

[13] Potdar, K., Pardawala, T.S. and Pai, C.D., 2017. A comparative study of categorical variable encoding techniques for neural network classifiers. International journal of computer applications, 175(4), pp.7-9.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., and Duchesnay, E. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, p.2825–2830.

[15] Wold, S., Esbensen, K. and Geladi, P., 1987. Principal component analysis. Chemometrics and intelligent laboratory systems, 2(1-3), pp.37-52.

Dep. Variable:	roc_auc_scores	**R-squared:**	0.976
Model:	OLS	**Adj. R-squared:**	0.975
Method:	Least Squares	**F-statistic:**	750.8
Date:	Wed, 18 Nov 2020	**Prob (F-statistic):**	1.12e-30
Time:	14:15:16	**Log-Likelihood:**	120.07
No. Observations:	40	**AIC:**	-234.1
Df Residuals:	37	**BIC:**	-229.1
Df Model:	2		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
Intercept	0.6582	0.003	192.228	0.000	0.651	0.665
C(layers_length)[T.5]	-0.1528	0.004	-38.654	0.000	-0.161	-0.145
opt_func[T.SGD]	-0.0108	0.004	-2.734	0.010	-0.019	-0.003

Omnibus:	32.129	**Durbin-Watson:**	0.976
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	75.535
Skew:	-2.065	**Prob(JB):**	3.96e-17
Kurtosis:	8.317	**Cond. No.**	3.19

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Dep. Variable:	roc_auc_scores	**R-squared:**	0.969
Model:	OLS	**Adj. R-squared:**	0.967
Method:	Least Squares	**F-statistic:**	558.1
Date:	Mon, 23 Nov 2020	**Prob (F-statistic):**	5.34e-15
Time:	18:58:49	**Log-Likelihood:**	77.372
No. Observations:	20	**AIC:**	-150.7
Df Residuals:	18	**BIC:**	-148.8
Df Model:	1		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
Intercept	0.5936	0.002	352.339	0.000	0.590	0.597
C(model_type)[T.RandomForestClassifier]	0.0563	0.002	23.623	0.000	0.051	0.061

Omnibus:	6.891	**Durbin-Watson:**	1.497
Prob(Omnibus):	0.032	**Jarque-Bera (JB):**	4.326
Skew:	0.915	**Prob(JB):**	0.115
Kurtosis:	4.357	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[16] Abdi, H. and Williams, L.J., 2010. Principal component analysis. Wiley interdisciplinary reviews: computational statistics, 2(4), pp.433-459.

[17] Chollet, F., and others. (2015). Keras. https://keras.io.

[18] Wes McKinney 2010 . Data Structures for Statistical Computing in Python . In Proceedings of the 9th Python in Science Conference (pp. 56 - 61).

[19] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing 2016. . Jupyter Notebooks – a publishing format for reproducible computational workflows

[20] Hunter, J. 2007. Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), p.90–95.

[21] Scikit-learn.org. 2020. Post Pruning Decision Trees With Cost Complexity Pruning — Scikit-Learn 0.23.2 Documentation. [online] Available at: ¡https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py¿ [Accessed 20 November 2020].

[22] Kakade, S.M., Shalev-Shwartz, S. and Tewari, A., 2012. Regularization techniques for learning with matrices. The Journal of Machine Learning Research, 13(1), pp.1865-1890.

[23] Kukačka, J., Golkov, V. and Cremers, D., 2017. Regularization for deep learning: A taxonomy. arXiv preprint arXiv:1710.10686.

[24] Team, K., 2020. Keras Documentation: Model Training Apis. [online] Keras.io. Available at: ¡https://keras.io/api/models/model_training_apis/¿ [Accessed 20 November 2020].

[25] Gal, Y. and Ghahramani, Z., 2016. A theoretically grounded application of dropout in recurrent neural networks. In Advances in neural information processing systems (pp. 1019-1027).

[26] Seabold, S., and Perktold, J. 2010. statsmodels: Econometric and statistical modeling with python. In 9th Python in Science Conference.

[27] Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1(5), pp.206-215.

	model_type	mean_roc_auc_scores	std_roc_auc_scores	upper_conf_roc_auc_scores	lower_conf_roc_auc_scores
0	DecisionTreeClassifier	0.593600	0.000000	0.593600	0.593600
1	RandomForestClassifier	0.649884	0.007534	0.663894	0.635875

	model_type	mean_f1_scores	std_f1_scores	upper_conf_f1_scores	lower_conf_f1_scores
0	DecisionTreeClassifier	0.560748	0.0000	0.593600	0.593600
1	RandomForestClassifier	0.600149	0.0183	0.663894	0.635875

	model_type	mean_r2_scores	std_r2_scores	upper_conf_r2_scores	lower_conf_r2_scores
0	DecisionTreeRegressor	0.334654	0.00000	0.334654	0.334654
1	RandomForestRegressor	0.109266	0.00355	0.115867	0.102665

	model_type	mean_mse	std_mse	upper_conf_mse	lower_conf_mse
0	DecisionTreeRegressor	6.852948	0.000000	6.852948	6.852948
1	RandomForestRegressor	9.174406	0.036563	9.242392	9.106420

Dep. Variable:	r2	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	4.031e+04
Date:	Mon, 23 Nov 2020	**Prob (F-statistic):**	1.30e-31
Time:	19:51:17	**Log-Likelihood:**	92.423
No. Observations:	20	**AIC:**	-180.8
Df Residuals:	18	**BIC:**	-178.9
Df Model:	1		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
Intercept	0.3347	0.001	421.600	0.000	0.333	0.336
C(model_type)[T.RandomForestRegressor]	-0.2254	0.001	-200.780	0.000	-0.228	-0.223

Omnibus:	7.503	**Durbin-Watson:**	1.969
Prob(Omnibus):	0.023	**Jarque-Bera (JB):**	6.107
Skew:	-0.641	**Prob(JB):**	0.0472
Kurtosis:	5.384	**Cond. No.**	2.62

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Dep. Variable:	r2	**R-squared:**	1.000
Model:	OLS	**Adj. R-squared:**	1.000
Method:	Least Squares	**F-statistic:**	2.767e+05
Date:	Mon, 23 Nov 2020	**Prob (F-statistic):**	5.82e-78
Time:	20:01:08	**Log-Likelihood:**	206.05
No. Observations:	40	**AIC:**	-406.1
Df Residuals:	37	**BIC:**	-401.0
Df Model:	2		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
Intercept	0.3423	0.000	857.716	0.000	0.341	0.343
C(layers_length)[T.5]	7.323e-05	0.000	0.159	0.875	-0.001	0.001
opt_func[T.SGD]	-0.3428	0.000	-743.925	0.000	-0.344	-0.342

Omnibus:	50.424	**Durbin-Watson:**	1.495
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	344.539
Skew:	2.772	**Prob(JB):**	1.53e-75
Kurtosis:	16.266	**Cond. No.**	3.19

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified