Multi-Class Classification of Primary Tumor Sites Using Supervised Machine Learning Methods

Brian Ripley, Andrew Hernandez, Rodas Hailu

University of California, San Diego

Abstract

In this paper, several supervised machine learning models are explored and their accuracies compared. The models are implemented in an attempt to classify the primary tumor location given the anatomical regions of the body which the cancer has spread. Each of the four models were similar in performance, with the logistic regression being prominent during the cross-validation, and the support-vector machine during testing. These models use limited data to make predictions that are far more reliable than a random guess given the number of unique targets in the dataset.

Introduction

Approximately 3% to 5% of cancer diagnoses are cancer of unknown primary (CUP). The prognosis for a patient diagnosed with CUP is generally very poor with near ensured morbidity. The issue surrounding CUP is that, at time of discovery, malignant cells are present in multiple areas of the body [1]. This creates an issue for treating the cancer as treatment usually targets the primary tumor location. In some cases of CUP the primary location will never be identified and even postmortem, the origin is unidentifiable [1]. This manifests the challenge in identifying the primary location as quickly as possible in order to begin treatment. Often, a medical professional will make an educated guess as to where the cancer originated [2].

A predictive model to identify the origin would prove to be very useful in the medical field to improve the prognosis for the unfortunate who are diagnosed with this cancer that is relatively high in both incidence and mortality [1]. Utilizing data from patients with known primary, along with the anatomic regions where the cancer has metastasized, it may be possible to predict the primary location in CUP patients when the areas of metastization are known.

This project makes use of the Primary Tumor Dataset. Provided in the dataset are the areas of the body in which the cancer has metastasized. The intent is to identify where the primary tumor is located from the areas where it has metastasized. Other features that will be used to increase the accuracy of the model include the age and sex of the patient, the histologic type of the tumors, and the degree of differentiation of the malignant cells.

Problem solving Approach

The data set was picked from the UCI repository online database platform specific on primary tumors. A Gridsearch was used to search the space of hyperparameters for each one of four models in order to find the optimal combination that leads to the highest K-fold cross validation score using the training data. The four supervised learning methods we chose to use consisted of a logistic regression, support-vector machine, K-nearest neighbors, and random forest. Running each model and getting a training and test set performance score would be considered one trial, and we'd do this ten times to get an average more representative of the model's performance. Throughout this process it was difficult to stratify the split of the data and ensure that there was a proportional amount of target labels in both the training and test sets, since many classes have less than 3 examples and this makes it difficult to have a limited number of examples exist in multiple splits of the data. This issue would be accounted for by imposing constraints on the number of possible predictions the model can as a whole. We'd remove all instances of classes that have less than 10 examples in order to ensure that the model is only making predictions using categories it has learned from a larger number of examples.

Solutions

The first steps before any processing was performed involved checking for missing data and deciding how to deal with it. The data set consists of 18 attributes: ['age', 'sex', 'histologic-type', 'degree-of-diffe', 'bone', 'bone-marrow', 'lung', 'pleura', 'peritoneum', 'liver', 'brain', 'skin', 'neck', 'supraclavicular', 'axillar', 'mediastinum', 'abdominal', 'class']. For classes where a large number of missing values are present, we decided to use the mode of the feature vector/column to fill missing values. Though this is not necessarily good practice, as we are tinkering the data, it was found to produce better accuracy than other strategies for dealing with missing data. Normally, the most optimal method, given that the dataset is large enough, would be to drop the data with missing attributes. However, we did not want to drop so much data when the data set was already fairly small.

This strategy was used for the degree of differentiation column. The degree of differentiation of a tumor is a description of the degree to which a tumor cell's appearance is similar to that of the tissue in the region it is occupying. The mode of this attribute, by a hefty margin, was poorly differentiated. That is, the cancer cells are less similar in appearance to normal tissue. As the majority of tumors fall into this category, it somewhat justified the use of this value in place of the missing data [3].

This method was also attempted for the histologic type attribute. Besides primary site (the target class in this project), histologic type is the other main property in classifying cancers. The histologic type data was more diverse and the mode did not account for the majority of the other data, unlike the degree of differentiation. For these reasons, and because histologic type is an important feature in accurately identifying the primary site, the decision was made to drop the rows in which the histologic type was not reported [4].

None of the data in the dataset was numerical. Therefore, data preprocessing involved encoding categorical inputs into either ordinal or non-ordinal numerical equivalents. One hot encoding was necessary to do so. The first step is integer encoding. Each categorical attribute is transformed from text to an integer value that will represent the category value. This is crucial preprocessing of data in machine learning. For the categorical value to be represented as an integer allows for the model to interpret a relationship through the natural order of numbers [5]

This would be sufficient for certain variables in which an order is inherent. For instance, placing in a race: first, second, third. In our data, the only variable we wanted to encode ordinally is the age of the patient. So we left that data simply encoded as integers. However, this is not the case for most of our data. It would be negligent to leave our data encoded as integers as the model could interpret a nonexistent ordered relationship, which in turn could output whacky results and poor accuracy. Thus, one hot encoding removes this relationship, making the integer numbers binary variables [5]. This new representation does not assume any order in the attributes.

```
Out[14]: array([[1.0, 0.0, 0.0, ..., 1, 0, 0],
                [1.0, 0.0, 0.0, ..., 0, 0, 0],
                [1.0, 0.0, 0.0, ..., 0, 0, 0],
                ...,
                [0.0, 0.0, 1.0, ..., 1, 0, 0],
                [1.0, 0.0, 0.0, ..., 0, 0, 0],
                [1.0, 0.0, 0.0, ..., 0, 0, 1]],
```

Figure 1 : The encoded data after one-hot encoding

Our particular dataset has a large number of target classes, with some having less than two instances. Several strategies were tested to account for this issue. Of course, we initially spent a great deal of time searching for supplementary data, but to no avail. Another strategy was to group the target classes of similar tissue type together so that there were greater amounts of instances for each target, though this would take away from the overall motivation for such a project, as it is necessary to know the exact location of the primary tumor. We also tested our models on just the top five most common target classes. Because this strategy greatly decreased the amount of data in the dataset, we finally decided we would drop classes that have a total number of examples less than some arbitrary integer. This allows the model to be trained to make predictions on classes that have a good enough number of useful examples. The end result will be a

model that can still help make meaningful predictions, but it's predictions will be limited to classes that are chosen to be kept.

The decision was made to drop all classes with less than ten instances. This reduced the number of target classes from 21 to 10 while only sacrificing 45 data points. Any removal of data during preprocessing was thoroughly mulled over due to the number of initial observations in the dataset being relatively small at just 329. If additional data were acquired, these classes would certainly be included in the data. However, in the interim, for the sake of the models' accuracy, it makes sense to focus on the target classes that are more prevalent in the dataset.

Due to the relatively small amount of data in our dataset, a simple train-test split will not suffice as this carries the potential for overfitting. K-fold cross validation was used to test the accuracy of each of our models because of its ability to reduce bias and gauge the overall generalizability of a machine learning model to unseen data. K-fold cross validation involves splitting the dataset into k subsets which will each have a turn as the testing data. The model is then tested k times with each of the k subsets being the testing data once, while the rest of the subsets serve as its training data [6].

After all the preprocessing of the data, we were ready to implement our models. A very useful function for the implementation of our models was Grid Search. Grid Search proves to be particularly useful in saving time and effort when dealing with imbalanced target classes and small sample size, just as we are in this project. Grid Search takes an input dictionary with the hyperparameter values to be tested and returns the most optimal combination of those hyperparameters for the particular training data and model [7]. In doing so, it saves a great amount of time in manually tuning the hyperparameters when the optimal values will differ across the training sets.

```
[96]
space = {'C': [0.001, 0.01, 0.1, 1, 10, 100], 'gamma': [0.0001, 0.001, 0.01, 0.1], 'kernel': ['rbf',
'linear']}
search = GridSearchCV(model, space, scoring = 'accuracy', n_jobs = -1, cv = cv)
result = search.fit(X_train, y_train)
print('Best Hyperparameters: %s' % result.best_params_)

Best Hyperparameters: {'C': 1, 'gamma': 0.0001, 'kernel': 'linear'}
```

Figure 2: An example of Grid Search being run on one of the models

For each of the four methods used in this project to identify the primary tumor location, it is important to understand the motivation behind their use. We will begin with logistic regression, which is a classification algorithm that is designed for binary problems, but with modification may be used for a multi-class classification. In order to use logistic regression to work with multi-class data, the multi-class data needs to be split up into multiple binary problems. We used the one-vs.-rest strategy. which creates a single binary classification per class [8]. In other words, as the name implies, each binary classification is one class vs. all of the other classes. Thus, for the ten classes in our data, ten binary classification problems are produced. A probability is predicted from each model, and the class of argmax will be the predicted class for that testing data.

```
Binary Classification Problem 1:
breast vs. [ovary, colon, lung, head and neck, pancreas, kidney, gallbladder, stomach, thyroid]
Binary Classification Problem 2:
ovary vs. [breast, colon, lung, head and neck, pancreas, kidney, gallbladder, stomach, thyroid]
Binary Classification Problem 3:
colon vs. [breast, ovary,  lung, head and neck, pancreas, kidney, gallbladder, stomach, thyroid]
Binary Classification Problem 4:
lung vs. [breast, ovary, colon, head and neck, pancreas, kidney, gallbladder, stomach, thyroid]
Binary Classification Problem 5:
.
.
.
```

Figure 3: One-vs.-Rest binary classification problems produced from the multi-class data

The second method we used to classify primary tumor locations was k-nearest neighbors (kNN). Unlike logistic regression, it is designed for multi-class classification, so is actually better suited for our project. kNN utilizes a measure of distance which could be Euclidean, Manhattan, Minkowski, Cosine, etc. Due to the use of Grid Search, the hyperparameter "metric" in the kNN classifier may be any of these. kNN looks at the k closest data points which are decided by one of the previously listed metrics of distance. Each metric may produce different nearest neighbors, thus, the Grid Search will sometimes use different measures of distance for that parameter. However, Minkowski distance seemed to be most commonly used:

$$\mathrm{dist}(\mathbf{x}, \mathbf{z}) = \left(\sum_{r=1}^{d} |x_r - z_r|^p \right)^{1/p}. \quad (1)$$

This returns the distance between point x and point z where p is the order of the norm. For purposes of familiarity, Minkowski distance is actually a generalized form of Euclidean distance, which is also a possible metric of distance used for KNN [9]. When p is equal to 2 in equation (1), we get the equation for Euclidean distance (note p below is a separate variable from equation (1)):

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \quad (2)$$

After kNN utilizes one of the distance measures to determine the k nearest neighboring training data points, it uses a majority vote to decide what the predicted class will be. That is, if three out of five of the nearest neighbors were 'lung' and the remaining two were 'thyroid', the predicted class would be 'lung', as it received the majority of the votes. kNN is sometimes referred to as a "lazy" algorithm considering it does not actually make any models [10]. It is also, computationally, a burden as it needs to compute a great number of distances for a large dataset. However, it is well-suited for the project at hand in which we have a rather small dataset and a multi-class classification.

Another important parameter for kNN is the number of nearest neighbors that the algorithm will look at when predicting an unseen datapoint. The number of nearest neighbors in our algorithm will vary due to Grid Search, just as the distance measurement. However, when analyzing each run of the algorithm it was important to note the number of neighbors that were used. Smaller values of k can lead to underfitting while larger values of k may result in overfitting [11]. This is a result of the bias-variance tradeoff.

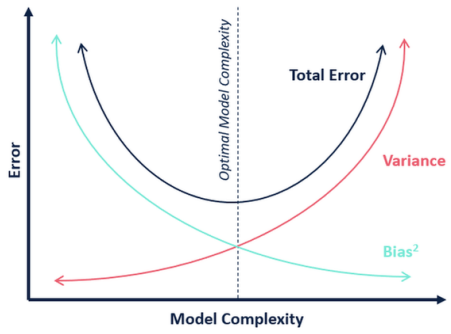


Figure 4: The Bias-Variance Tradeoff [12]

As bias decreases with a greater value for k, variance decreases. Therefore, when choosing k we need to be cognizant of both bias and variance such that the error of predictions is at its minimum. The bias-variance tradeoff is not just important for determining the value for k, but is very important in model selection. The best machine learning model for a problem will not be under- or overfitting for the training data in order for it to be accurate when unseen data is used. When running Grid Search on our kNN model, the k-value ranged from 4 to 15. In a finalized model, we would choose a value somewhere in the middle of this range to ensure bias and variance are, together, minimized.

Our third method for classification is a random forest. Before tackling a random forest model, we need to understand decision trees. A decision tree branches out based on attributes until it reaches a conclusion. For our data, a decision tree may check whether the cancer has metastasized to each body area in the data set. It would continually branch for each attribute it encounters until reaching the target class that meets the requirements.

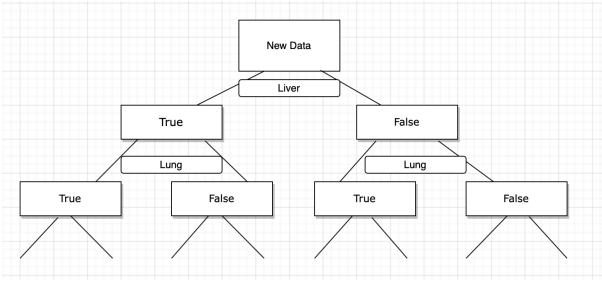


Figure 5: An example of the beginning of a decision tree that may be used in the random forest model for our data

A random forest makes use of multiple decision trees as seen in figure (5). These decision trees may reach different conclusions, so majority voting (as discussed with kNN) decides which class will actually be predicted. Two important hyperparameters for the random forest model are "n_estimators" and "max_features." The

former indicates how many decision trees to be run and the latter indicates how many features will be considered at each node of those decision trees. For instance, in figure (5), “max_features” is equal to one. Increasing “max_features” generally increases the accuracy of individual trees, but removes diversity among the trees [13]. Therefore, similar to balancing bias and variance when determining the value of k in kNN, “max-features” needs to balance the individual performance of trees and the diversity across trees. If we were to focus on the individual performance of trees, we would run the risk of overfitting the model. When running the model with Grid Search a number of times, the value used for “max_features” ranged from 1 to 16. In a finalized kNN model, we would want to use a number somewhere in the middle of this range.

The final method we used to predict primary tumor location was support vector machines (SVM). A very important characteristic of SVM is its tendency of not overfitting, so it generally performs well on unseen data. This may be the reason that SVM turned out to have the best accuracy on testing data. Like logistic regression, SVM is designed for binary classification. However, approaching it similarly to logistic regression, it can be used for multi-class classification through modification. As we did with logistic regression, the data set is broken down into multiple binary problems using the one-vs.-rest strategy (see figure(3)).

In SVM, to separate the data points, a hyperplane is created in n-dimensional space to split the target classes [14]. The shape of this hyperplane is determined by the kernel hyperparameter. Our Grid Search only looks at linear and radial basis functions (rbf) for the kernel type. We chose linear because it is best for data with many features, such as our data, and rbf because it is generally most preferred. Our Grid Search returned each of these approximately the same portion of the time. Therefore, it would require further testing to conclude which we would use for a final model.

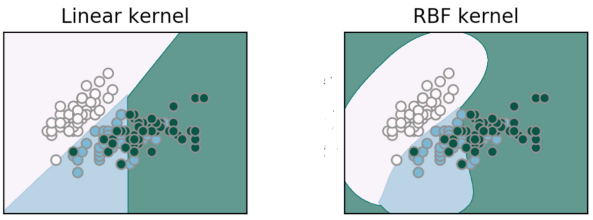


Figure 6: An example of a linear kernel vs. an rbf kernel [14]

This figure only represents two features, but for our data, these two dimensional separations would be hyperplanes in a high dimensional space. For a large number of features, one can see how a linear kernel may be better, for it will be more generalizable to new data. Rbf maintains more specificity, so it carries the possibility for overfitting.

The hyperparameter C will also impact how the hyperplane is situated. C determines how specific the hyperplane is formed for the training data. A high value of C will result in overfitting [9]. The Grid Search often used a C value of one, which is the default value. The hyperparameter gamma is used when the kernel type is rbf:

$$k(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2) \quad (3)$$

The rbf function changes depending on the distance from a location, producing the smooth curves as seen in figure (6). Like C, gamma needs to balance how well the hyperplane will split the target classes in the training data, and how well it generalizes to unseen data [9]. Rbf is very sensitive to gamma. Large gamma values will result in great overfitting and too small gamma values will result in poor placement of hyperplanes. The gamma value from Grid Search was typically 0.001 or 0.01. This is on the smaller end, but there are many features in our dataset and the dataset is small, so it will prevent overfitting.

As we are fairly new to machine learning, we tested all four of these models on our dataset: two algorithms that are regularly used for multi-class classification and two algorithms that required

some modification. The models still need some tuning to create final models that could be used on validation data of unknown class. However, at this point, we have a good sense of the hyperparameters that would be used.

Conclusion/Future Directions

The logistic regression had the highest performance on the training set based on an accuracy metric with around 40% accuracy. Table 1 contains mean performance on the training set; the entry in bold is the accuracy belonging to the best performing model, and those labeled with asterisks could not be deemed significantly different in performance compared to the one in bold. Power values for the two-sample t-tests for each combination are displayed in table 3 and table 4 in the Appendix. While the logistic regression outperformed all others on the training set, the support vector machine proved to be the best when generalizing it's predictions to the test set with around 45.3% accuracy. The logistic regression was not significantly different from the support-vector machine and thus we could not reject the null hypothesis of identical expected values for test scores. Table 2 contains each model's mean performance on the test set. While the accuracy is below 50%, it's important to consider that there are a total of ten possible classes that can be predicted, so our model is much more reliable than a random guess that would have around a 10% chance of being correct.

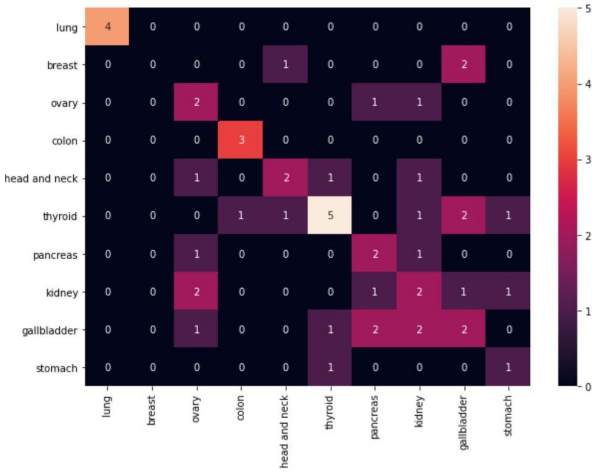
Column1	Mean Test Accuracy
Log	0.427*
K-NN	0.396
RF	0.41
SVM	**0.453**

Table 1: Mean Training Set Performance

Column1	Mean Training Accuracy
Log	**0.401**
K-NN	0.355
RF	0.397*
SVM	0.384*

Table 2: Mean Test Set Performance

For each model's predictions on the test set, we computed a confusion matrix to better understand which classes are being predicted as others mistakenly by our model. This is represented with a matrix with the same target classes on both x and y axises. A perfect model will only have entries on the diagonal, since this would mean each class is being predicted perfectly without error. Entries not on the diagonal show a mismatch between the ground truth and what the model predicts. Figure shows the confusion matrix for the best(above) and worst performing models on the test set, and it's clear that the SVM has more entries marked on the diagonal.



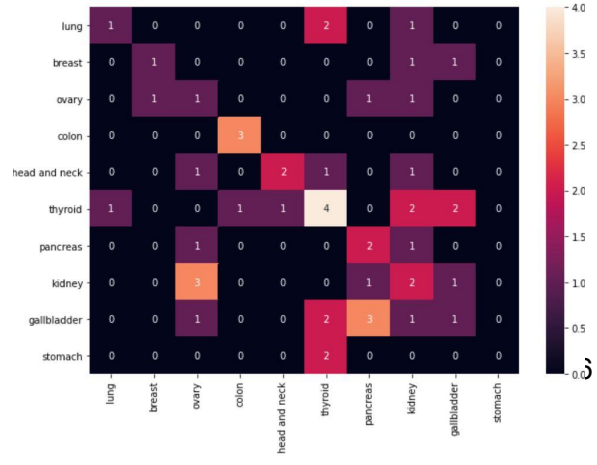


Figure 7. Confusion Matrix for test performance. Best model is the support vector machine above, and the worst performing model on average is K-nearest neighbors below.

Limitations in this dataset are prevalent with the few observations and missing data. Additional data sets would be ideal for further accuracy. Our initial project plan was to identify brain tumors from MRI or CT scans. In the future, this could be navigated using CNN or further, using deep learning processes to develop 3D models visualizing tumors. For future work, we can also consider using an image dataset of MRI or CT scans for tumor detection. It can be navigated using CNN or furthermore using a deep learning process inorder to develop 3D models of visualizing tumors.

Code Availability

All code used to run the four supervised models can be found in the linked github repository. The repository also contains the notebook used to perform the two-sample t-tests on the raw training and test accuracy scores.

References

[1] Dyrvig, Anne-Kirstine, et al. "Cancer of Unknown Primary: Registered Procedures Compared with National Integrated Cancer Pathway for Illuminating External Validity." *Medicine*, vol. 96, no. 16, Apr. 2017, p. e6693. *PubMed*, doi:10.1097/MD.0000000000006693.

[2] Qaseem, Aisha et al. "Cancer of Unknown Primary: A Review on Clinical Guidelines in the Development and Targeted Management of Patients with the Unknown Primary Site." *Cureus*vol. 11,9 e5552. 2 Sep. 2019, doi:10.7759/cureus.5552

[3] Jögi, Annika, et al. "Cancer Cell Differentiation Heterogeneity and Aggressive Behavior in Solid Tumors." *Upsala Journal of Medical Sciences*, vol. 117, no. 2, May 2012, pp. 217–24. *PubMed*, doi:10.3109/03009734.2012.659294.

[4] Girard, Nicolas, et al. "Comprehensive Histologic Assessment Helps to Differentiate Multiple Lung Primary Nonsmall Cell Carcinomas from Metastases." *The American Journal of Surgical Pathology*, vol. 33, no. 12, Dec. 2009, pp. 1752–64. *PubMed*, doi:10.1097/PAS.0b013e3181b8cf03.

[5] Brownlee, Jason. "One-vs-Rest and One-vs-One for Multi-Class Classification." Machine Learning Mastery,https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

[6] Tabe-Bordbar, Shayan, et al. "A Closer Look at Cross-Validation for Assessing the Accuracy of Gene Regulatory Networks and Models." *Scientific Reports*, vol. 8, no. 1, Apr. 2018, p. 6620, doi:10.1038/s41598-018-24937-4.

[7] B. H. Shekar and G. Dagnew, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data," 2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), 2019, pp. 1-8, doi: 10.1109/ICACCP.2019.8882943.

[8] Brownlee, Jason. "One-vs-Rest and One-vs-One for Multi-Class Classification." Machine Learning Mastery,https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

[9] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

[10] M. Zhang, Z. Zhou. "ML-KNN: A lazy learning approach to multi-label learning." *Pattern Recognition,* vol. 40, no. 7, 2007, pp. 2038–48. https://doi.org/10.1016/j.patcog.2006.12.019.

[11] R.C. Neath and M.S. Johnson, "Discrimination and Classification." *International Encyclopedia of Education (Third Edition),* 2010, pp. 135-141, https://doi.org/10.1016/j.patcog.2006.12.019.

[12] Huilgol, Purva. "Bias and Variance Tradeoff: Beginners Guide with Python Implementation." *Analytics Vidhya*, 16 Aug. 2020, www.analyticsvidhya.com/blog/2020/08/bias-and-variance-tradeoff-machine-learning/.

[13] Srivastava, Tavish. "Random Forest Parameter Tuning: Tuning Random Forest." *Analytics Vidhya*, 26 June 2020, www.analyticsvidhya.com/blog/2015/06/tuning-random-forest-model/.

[14] Marius, Hucker. “Multiclass Classification with Support Vector Machines (SVM), Kernel Trick & Kernel Functions.” *Medium*, Towards Data Science, 9 Sept. 2020, https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02.

Appendix

Table 3: p-values from a two-sample t-test using test set accuracy

Column1	Test Set p-values
Log. vs K-NN	0.181396517
Log vs RF	0.443040742
Log vs SVM	0.225367828
K-NN vs SVM	0.02201962
K-NN vs RF	0.584853499
RF vs SVM	0.072630967

Table 4: p-values from a two-sample t-test using training set accuracy

Column1	Training Set p-values
Log. vs K-NN	0.000122547
Log vs RF	0.623585492
Log vs SVM	0.082350992
K-NN vs SVM	0.002203133
K-NN vs RF	0.000126314
RF vs SVM	0.155124667