Exploring the Error vs. Complexity Tradeoff for Different Classification Models for Diabetes 130-US Hospitals

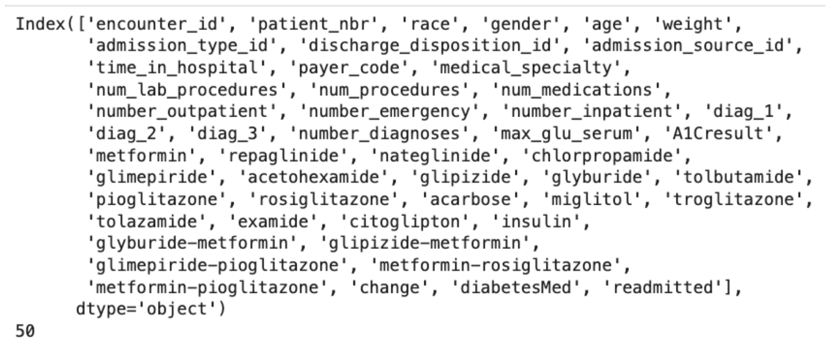
Names: Kayla Kim (49088756), Rosnita Dey (77717795), Jayson Nguyen (59406301)
Github Repository: https://github.com/kimkc1/Diabetes-Classifier.git

Summary

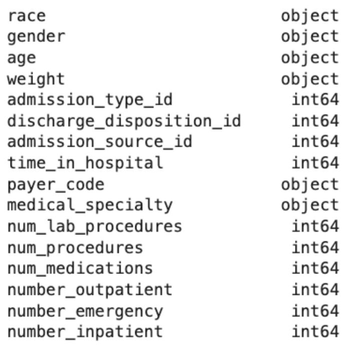
Our project explores error versus complexity tradeoffs for different classification models using the Diabetes 130-US Hospitals dataset. We investigated the efficacy of different classification methods including logistic regression, decision trees, K nearest neighbors, and neural networks, on predicting readmission results within 30 days for patients with diabetes. Our experiments on these different classifiers showcased remarkably similar error rates, around 40% for each model that was tested.
demonstrated very similar error rates or around 40% for each model that was tested. In general, we discovered that as hyperparameters are adjusted and the complexity of the model increases, the error rate on the training dataset decreases while the error rate on the testing dataset increases. Overfitting on the training dataset most likely causes this increase in testing error.

Data Description

The Diabetes 130-US Hospitals dataset is structured in tabular format, with each instance representing hospital records of patients diagnosed with diabetes. These records are mapped to a target variable representing hospital readmission after 30 days. The dataset also consists of 49 features, encompassing both numerical and categorical types. The following figures display each of the feature names contained in the dataset as well as the data type for the first few features.

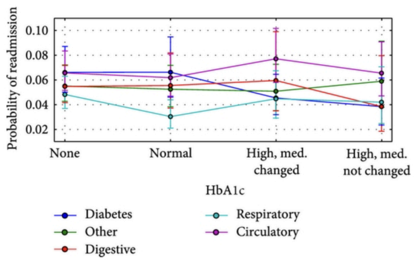


```
Index(['encounter_id', 'patient_nbr', 'race', 'gender', 'age', 'weight',
       'admission_type_id', 'discharge_disposition_id', 'admission_source_id',
       'time_in_hospital', 'payer_code', 'medical_specialty',
       'num_lab_procedures', 'num_procedures', 'num_medications',
       'number_outpatient', 'number_emergency', 'number_inpatient', 'diag_1',
       'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult',
       'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
       'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
       'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
       'tolazamide', 'examide', 'citoglipton', 'insulin',
       'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-rosiglitazone',
       'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted'],
      dtype='object')
50
```



```
race                          object
gender                        object
age                           object
weight                        object
admission_type_id              int64
discharge_disposition_id       int64
admission_source_id            int64
time_in_hospital               int64
payer_code                    object
medical_specialty             object
num_lab_procedures             int64
num_procedures                 int64
num_medications                int64
number_outpatient              int64
number_emergency               int64
number_inpatient               int64
```

In the research article “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records”, the diabetes dataset is used to determine the probability of readmission based on multiple features. The article uses age to look at readmission rates, but it also uses the primary diagnosis and HbA1c measurement. The figure below shows the findings from the article on the probability of readmission based on the primary diagnosis and the Hb1Ac measurement (“Impact of HbA1c Measurement”).



Classifiers

Logistic Regression

Logistic Regression is a statistical model used to find the probability of a binary event occurring. The model uses a logistic function, to transform linear combinations of input features into values between 0 and 1, which are interpreted as probabilities. To implement a logistic regression model we used Scikit-learn's Logistic Regression and selected different hyperparameters to optimize its performance. We tested different values for C, the hyperparameter that controls regularization strength, within the range [0.001, 50]. We also adjusted the penalty, which specifies the normalization factor, between 'l1' (Lasso) and 'l2' (Ridge). The max_iter parameter specifying the maximum number of iterations taken for the solver to converge was also tested using values using values in the range [100, 1000].

K Nearest Neighbor

K Nearest Neighbor is a supervised learning classifier that uses proximity to make predictions about the grouping of an individual data point. This classifier works on the assumption that there are similar points grouped around one another. Given a data point and k neighbors, we find the k closest points to the data point and decide the label based on the majority. The hyperparameter we manipulate in this classifier is k, the number of nearest neighbors, which ranged from [1, 215] in our experiments. To implement this model, we used the K Nearest Neighbor classifier offered by the Scikit-learn library.

Decision Tree

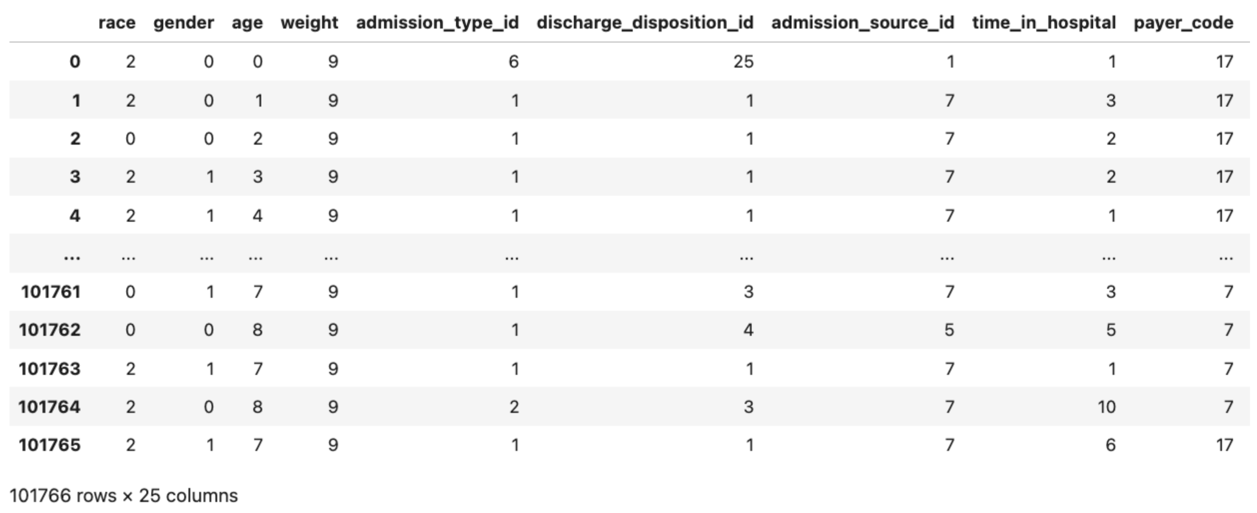
The Decision Tree classifier is a supervised learning method that learns simple decision rules from the data. A tree structure is created, where each internal node represents a feature, each branch represents a decision based on a feature, and each leaf node represents an outcome. The hyperparameters used for the decision tree were max_depth and min_samples_leaf. The max_depth parameter is the maximum depth of the tree, and the min_samples_leaf parameter is the minimum number of samples required to be at a leaf node. For both hyperparameters, we investigated the range [1, 50]. We used the DecisionTreeClassifier offered by the Scikit-learn library.

Neural Networks

Neural Networks represent a computational model inspired by the structure and function of biological neural networks. Composed of interconnected nodes organized in layers, neural networks typically include an input layer, one or more hidden layers, and an output layer. Each node is linked to others with associated weights and thresholds. The hyperparameters we adjust are the number of hidden layers, which range from [32, 512], and the learning rate for values within the range [0.0005, 0.5].

Experimental Setup

Since our dataset consists of non numerical features and missing values, we had to preprocess the data before feeding it into our models. We went through our data and dropped features that were missing 60% or more missing values. Label Encoder was then used to convert any categorical or non numerical values to integer values. For the remaining columns that contained missing values, we took the mean of that feature and used it as a placeholder value. The following figure shows the first few rows and columns for our dataset after processing.



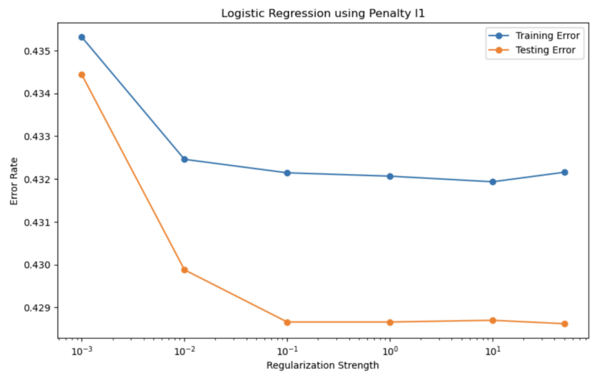
	race	gender	age	weight	admission_type_id	discharge_disposition_id	admission_source_id	time_in_hospital	payer_code
0	2	0	0	9	6	25	1	1	17
1	2	0	1	9	1	1	7	3	17
2	0	0	2	9	1	1	7	2	17
3	2	1	3	9	1	1	7	2	17
4	2	1	4	9	1	1	7	1	17
...	...	...	...	...	...	...	...	...	...
101761	0	1	7	9	1	3	7	3	7
101762	0	0	8	9	1	4	5	5	7
101763	2	1	7	9	1	1	7	1	7
101764	2	0	8	9	2	3	7	10	7
101765	2	1	7	9	1	1	7	6	17

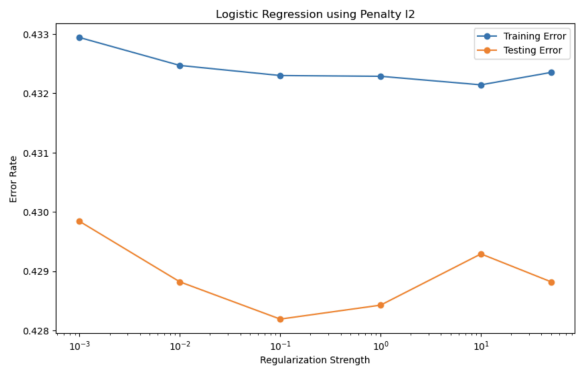
101766 rows × 25 columns

For testing and training, the dataset was partitioned into a 25-75 split, with 75% of the data being used to train and validate the model and the remaining 25% used to test and evaluate final model performance. For each model we looked at the metric classification accuracy, to assess the performance of each model under various sets of hyperparameters.

Experimental Results

Logistic Regression

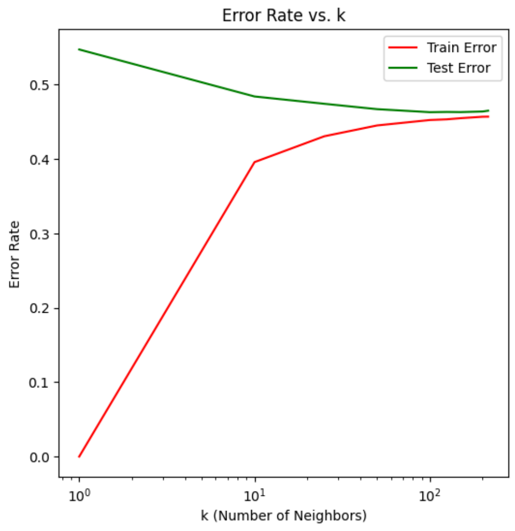




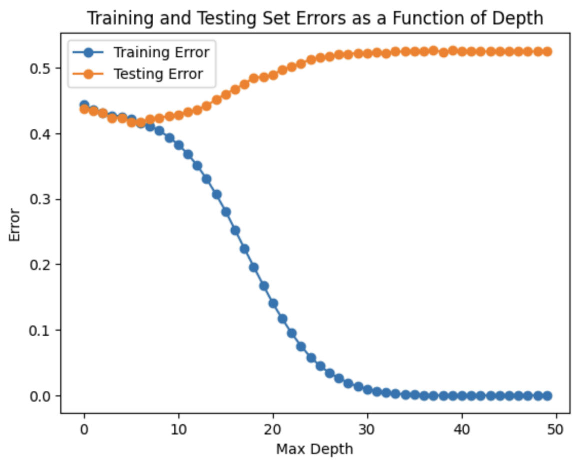
Our experimentation with logistic regression models revealed that increasing the regularization strength led to decreasing error rates on both the training and testing data sets. We observed this trend in decreasing error rate irrespective of whether we employed the L1 or L2 penalty. Although both error rates had a decreasing trend, we found that when using the L2 penalty, the error decreases at a much slower rate and the training error remains relatively high compared to the L1 penalty. Our logistic regression models had an average error rate of 0.43 or 43%. Increasing regularization prevents overfitting from occurring and favors simpler solutions, this is most likely why we observe a decrease in testing error. The plateauing error rate is most likely observed because the model has converged towards an optimal solution where further adjustments will lead to minimal improvement in performance.

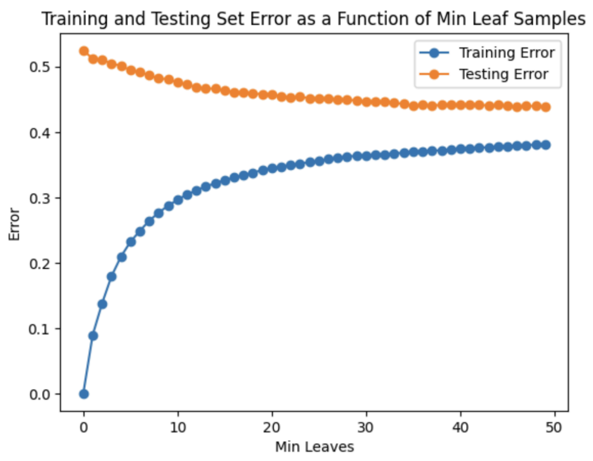
K Nearest Neighbor

For our kNN classifier, we varied the parameter, k, representing the number of nearest neighbors that each data point would take into consideration. Given the size of our dataset, we chose k values as high as 2000 in order to fully understand the impact of k on the classifier's error rates. We found that the testing and training errors tend to converge along a singular axis (around 45% error rate). As we keep increasing k neighbors, we can see that the graph tapers off and reaches about 45%. The convergence suggests that regardless of the number of neighbors considered, the classifier's performance stabilizes around this threshold.



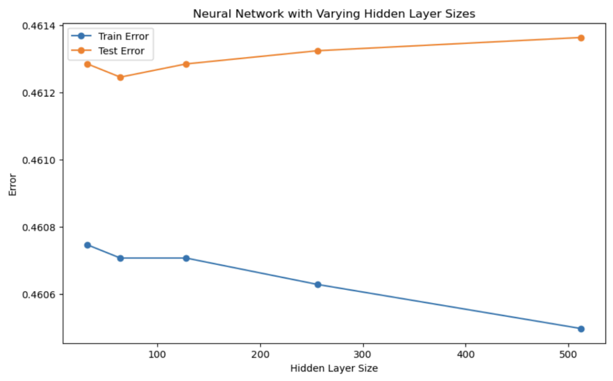
Decision Tree

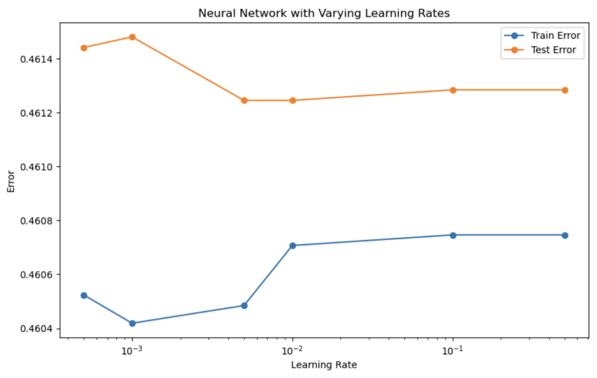




Regarding decision trees, we observed that as the maximum depth was increased, the error rate for the training set decreased towards zero. On the other hand, the error rate for the testing set displayed a small decrease before rising again and stabilizing at a 50% error rate. As the maximum depth of the tree increases, the model is able to capture more complex patterns, hence the decreasing error rate on the testing dataset. However, the testing error rate increases after a certain point due to overfitting. We also observed that when the minimum leaf samples were increased, the error rate for the training set increased logarithmically. However, the error rate for the testing set decreased steadily, and both appear to converge to an error rate of around 40%. The convergence suggests that the model has found an optimal balance between complexity and generalization and any further changes to the minimum leaf hyperparameter will not result in much change in error rate.

Neural Networks





For neural networks, we found that increasing the hidden layer size results in a decrease in training error, yet the test error began to rise after a certain threshold, indicating potential overfitting to the training data. This trend is most likely observed because increasing the hidden layer size allows the model to capture more complex relationships in the training data; the model begins to memorize noise in the training data after a certain point. Furthermore, when we increased the learning rate, we observed an initial decrease in training data error, followed by a subsequent increasing trend. Conversely, the testing data displayed an initial increase in error, followed by a gradual decrease. This stabilization suggests that the model has reached a performance plateau, where further adjustments will result in marginal improvements to the error rate. The error rate for all our neural networks seemed to level out around an error rate of 0.46 or 46%.

Insights

Overall, our findings suggest that as the complexity of a model increases, the testing error will typically increase as well due to overfitting on the training data. This observation emphasizes the balance between model complexity and generalization performance; if a model memorizes unnecessary noise in the training data, it may struggle to generalize well to unseen instances (the testing data in this case). Another trend we observed was an eventual plateauing in error rate after a certain threshold for hyperparameters such as minimum leaf samples in the decision tree classifier and the number of nearest neighbors in the K nearest neighbors classifier. Recognizing this point of convergence can help identify an optimal level of complexity for our models and avoid the unnecessary computational expenses linked to hyperparameter fine-tuning. Despite differences in implementation, most of our classifiers have a similar error rate of 40-50%. This consistency in error rate suggests that our classifiers are robust and reliable with each classifier able to capture the underlying patterns in the diabetes dataset. However, out of all of our models, logistic regression using the L1 penalty resulted in the lowest testing error rate. This suggests that logistic regression is particularly adept at capturing nuances in datasets containing many features.

Contributions

- ***Jayson Nguyen*****:** I primarily worked on testing the K Nearest Neighbors classifier. I also wrote in our exploration setup how we were testing our dataset.
- ***Rosnita Dey:*** I worked on explaining and testing the Decision Tree Classifier, and I also wrote about and cited the paper that used the diabetes set in our data description.
- ***Kayla Kim:*** I worked on the Logistic Regression and Neural Networks Classifiers. I also worked on exploratory data analysis.

Citations

Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore, “Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records,” BioMed Research International, vol. 2014, Article ID 781670, 11 pages, 2014.