Saptarshi Ghose
Machine Learning in Cancer

Lab 4 Write-Up

In order to prepare the data that was used for modeling, I initially downloaded the provided lung-v03.txt data and stored it as a dataframe called *lung_df* in a Python Jupyter notebook. I proceeded by doing some initial cleaning of the county names so that just the county and state would remain in the 'County' column. I then downloaded the median income level data for all counties – where the dimensions were "US by county", "Income", "Median family income", and "All races". I read this data in as a separate dataframe that I called *county_income_df*. From here, I merged *lung_df* and *county_income_df* on "FIPS", in order to get matched cancer incidence and income data by county, and stored the merged dataset as a new dataframe called *merged_df.* At this point, I filtered *merged_df* so that it only relevant regression columns would remain (namely: 'Age-Adjusted Incidence Rate - cases per 100,000', 'County', and 'Value (Dollars)'. In order to properly represent the county field in the models I built, I then used the Pandas function *pd.get_dummies* to get unique dummy columns for each of the counties represented in the dataset – and store these as a new dataset called *counties_dummies*. The next step from here was to join *counties_dummies* to the existing *merged_df* dataframe – and then to drop the no longer needed 'County' field.

At this point, I isolate the label (in this case 'Age-Adjusted Incidence Rate - cases per 100,000') and isolated the feature set (in this case 'Value (Dollars)' + each of the dummy columns for county). I then used the sklearn function *train_test_split* to split the data into relevant training and testing sets (with a test size of 0.2).

After all the data was prepared for use in the models, I made use of four different linear models available within sklearn's linear_model package. These included BayesianRidge, LassoLars, TheilSenRegressor, and Linear Regression. I constructed a loop that modeled the underlying the data with each of the regression algorithms chosen – and recorded the resulting coefficients (clf.coef_), mean squared errors (mean_squared_error(y_test, y_pred), and r^2 values (r2_score(y_test, y_pred)) in order to have comparable evaluation metrics across the models tested.

As we were warned, the results showed relatively little predictive power across the board. The best performing model, relatively speaking, was a BayesianRidge model with an R^2 value of 0.19 and a Mean Squared Error of 256.33. LassoLars produced an R^2 value of 0 with a Mean Squared Error of 317.19. TheilSenRegressor produced an R^2 value of .17 with a Mean Squared Error of 263.58. And finally, LinearRegression produced an R^2 value of .17 as well -- but with a Mean Squared Error of 262.97.

In order to improve upon this model in the future, I would seek to use TPOTRegressor to see if the tool is able to find a model and parameter combination for the regression modelling that performs better in the evaluation metrics I used – namely R^2 and Mean Squared Error. A graphical approach could also be a good next step – so that we could visually see how the distribution of data falls around the regression lines.
