Daniel McKemie

July 11, 2024

DATA 71200: Project 4 - Final Paper

The dataset I used for the first project consisted of cancer patient data. This included cancer type, age at diagnosis, staging, malignancy, and biographic data. It was then joined with a table of information around their specific genetic mutations.
The goal of the prediction was to gain relational insights about tumor mutation and survival rate, in relation to the patient's other categorical data. This proved to be difficult due to the complexities and inconsistencies in the tumor mutation data compared to the rest of the data of each case.

The primary difficulty that arose from using this dataset were the mutation categories being a one-to-many relationship of patient to mutation information, resulting in a drastic increase in rows. I spent a lot of time seeing which types of joins and keys would lend themselves best to getting a proportionate dataset, but ultimately it was left with a lot of blank or duplicate rows. This made it particularly difficult to clean the data and lead to overfitting. The prediction goal was to determine which metric(s) were most closely associated with eventual survival or death, the OS_STATUS, which was deduced to a 0 or 1 boolean value of LIVING or DECEASED.

```
# Load the data
dataframes = load_data_from_github()

# Check if dataframes were loaded
if dataframes:
    df_patient = dataframes.get(DATA_FILE_NAME_PATIENT)
    df_patient.columns = df_patient.columns.str.upper() # Capitalize Column Nam

    df_sample = dataframes.get(DATA_FILE_NAME_SAMPLE)
    df_sample.columns = df_sample.columns.str.upper()

    df_gene_panel_matrix = dataframes.get(DATA_FILE_NAME_GENE_PANEL_MATRIX)
    df_gene_panel_matrix.columns = df_gene_panel_matrix.columns.str.upper()

    df_sv = dataframes.get(DATA_FILE_NAME_SV)
    df_sv.columns = df_sv.columns.str.upper()

    df_mutations = dataframes.get(DATA_FILE_NAME_MUTATIONS)
    df_mutations.columns = df_mutations.columns.str.upper()



    # Join tables
    merged_df = pd.merge(df_patient, df_sample, on="PATIENT_ID")
    merged_df = pd.merge(merged_df, df_gene_panel_matrix, on="SAMPLE_ID")

    # Left join to keep all patients
    # Right join to only keep patients who have gene/sample info
    merged_df = pd.merge(merged_df, df_sv, on="SAMPLE_ID", how="left")

    # A few dozen duplicates, arguably not needed.  Prune...
    merged_df.drop_duplicates(subset=['PATIENT_ID'], inplace=True)

    # Change variable name for clarity
    without_mutation_df = merged_df
```

Joining multiple tables

The lessons on data cleansing and data treatment were nonetheless incredibly valuable for this first project, even though the result could be seen as a "failure". To begin, the dataset was downloaded from the Memorial Sloan Kettering collections and provided in pieces. They had to be joined to assemble the full table. This was a great exercise in cleaning while joining, but also lead to some stalls in the process.

After joining the tables, I examined the best approach to deal with the excess from each table. I simply pruned categories with largely missing data that would not benefit by filling in with means, medians, or other recognizable consistencies. I believe that this was the biggest challenge because many of these columns, had there been more data and instances in general, would have likely provided valuable information for the predictions I was aiming for.

The next approach was filling in the data holes in the tumor counts categories, but the inconsistencies in the readings of each tumor to patient, and my lack of expertise on the matter, made it difficult to conclude anything. I went through the process of cleansing and visualizing for the sake of exercise, but it was then that I realized a new dataset would be necessary to achieve meaningful learning outcomes for the remaining projects. The decision was to pivot to a new dataset of baseball statistics for the second project. This gave a tighter group of categories, more rows of data, a basic time series (which, while not referenced in the final project submissions, did allow for some experimentation), and my own area of expertise, as I have better familiarity with baseball than advanced medicine.

The baseball data came largely cleansed, but I did some work on filling in gaps and adding one of my own categories based on the available data. The table consisted of rows of Major League Baseball team results by season, capturing total wins, runs for and against, batting statistics, rankings, and whether they made the postseason. It was

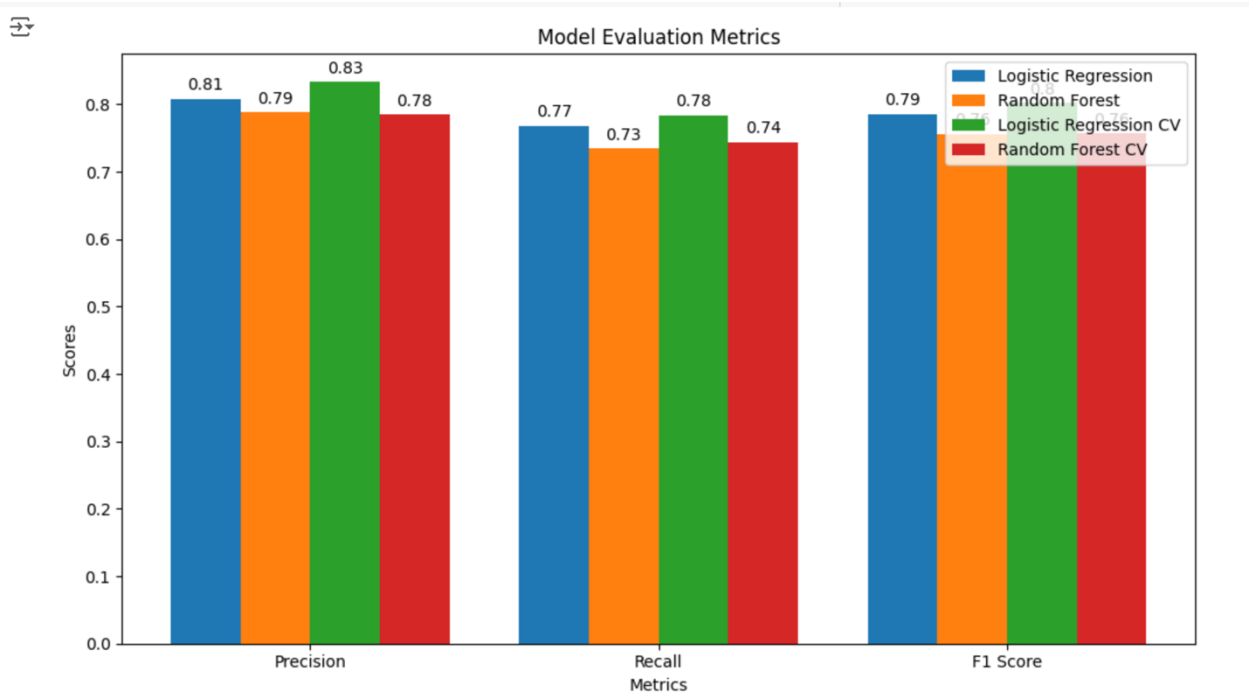
a relatively simple dataset that allowed for a straightforward prediction of, "What metric(s) contributed most to making the postseason?"

I added my own category of team OPS percentage, which is simply on-base plus slugging percentage. In the interest of time, I also dropped the overall season and playoff rankings categories. If more time had been allowed, I would have gathered other tables or scraped for standings data, as they would provide a great cross-reference for other statistics and tables. This would allow for more predictions and expand the model further.

The visualization of Project 1's data, which was the cancer dataset, did not yield much insight due to the problems mentioned above, other than overfitting was occurring. However, it was incredibly valuable for exploring how to visualize the set and gaining further familiarity with matplotlib. From this point on, I will refer to the lessons learned from the baseball dataset, used in the remaining projects. The visualizations in the remaining projects were very helpful as it gave a clear picture as to which algorithms were most effective and how, and which were not. I would like to take a more meaningful dataset back to the methods employed in Project 1 but have not yet done so.

The two supervised learning algorithms I chose to explore were Logistic Regression and the Random Forest Classifier. To begin with the first, it was the most obvious choice given the yes-or-no reality of a team making the playoffs based on the other categories of data. After working with this model, I was not entirely sure that it was the most

effective because the data was split by years. I believe this approach would have been even more effective in predicting whether a specific team was set to make the playoffs, given the season-by-season statistics in the dataset.



Visualization example of two approaches, with Random Forest being most effective

The best parameters for this algorithm as found by cross validation with GridSearchCV were:

- C = 1; This is the parameter to prevent large coefficients in the model to skew the result, so here 1 being the best param, was one option above the lowest (0.1)
- Penalty = 'l1'; The type of regularization applied, in this case the lasso option, adding the absolute value to the coefficient
- Solver = 'liblinear'; Given the dataset is small and liblinear the only option, it was left as such because it supports both l1 (Lasso) and l2 (Ridge) regularizations

The second algorithm was the Random Forest Classifier. This was chosen because the returned metrics were better performing than Logistic Regression. Perhaps also useful as it controls overfitting. For my best parameters, I would gather that the number of estimators being in the middle allowed for the greatest amount of growth and splitting without overfitting. Leading to the idea that the dataset, while simple, was of proper size? The three best parameters here were:

- n_estimators = 100; Total number of decision trees that are built, this is in the middle of the three options
- max_depth = None; The longest length from root to end leaf, in this case no maximum limit is set and the tree will grow until all leaves have the minimum samples required for splitting
- min_samples_split = 2; Allows splitting nodes with very few simples, having the highest chance of overfitting

PCA did not improve the results for either algorithm but also made them slightly less effective by about the same amount. After this project, I do wonder if using PCA properly would have improved my original cancer dataset. I have not yet taken the time to explore this but given the collection of information around this at the time of the paper, I would predict it to be at least somewhat the case. Also given this, I should have used the breast cancer data as instructed in the assignment to demonstrate proper pre-processing, as instructed, but did not.

```
from sklearn.preprocessing import StandardScaler

# Check and align the lengths of y and X_scaled
common_indices = X.index.intersection(y.index)
X_aligned = X.loc[common_indices]
y_aligned = y.loc[common_indices]

# Standardize the training data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

Pre-processing for PCA, but omitting other methods such as SimpleImputer, which should have been used

Throughout these projects and the class, I often felt somewhat lost during each current assignment right up until the end of said assignment. However, each subsequent assignment clarified even more about the previous one. Overall, it was a very rewarding, if somewhat stressful, experience. Summarizing the findings in this paper is encouraging and motivates me to continue exploring all of these tools and techniques with other datasets. I certainly feel prepared to employ any of them in my future work.

If I had to do it again, I would have been more focused and precise in using the cancer dataset, approaching the cleansing and categorizing in a different order. Additionally, I would have requested access to larger and more refined datasets through the CDC or NIH. However, given that these approvals take time for review, it was not a viable option for a summer course. I will certainly be exploring this in my own research for my thesis next year.