Lung Cancer Stage prediction with machine learning

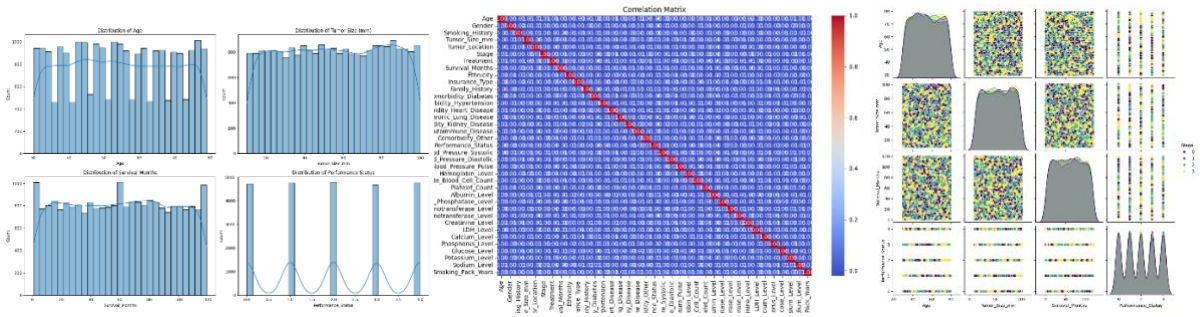
Vladimir Gološin, Kristina Andrijin, Stefan Bogdanović

1. Motivation

Doctors are in short supply and those who are available are often overworked and tired, especially the lung cancer specialists. In order to help them out a machine learning model can be created to aid in cancer diagnostics.

2. Research questions

Specifically, we want to make a machine learning model for classification of lung cancer stages based on various parameters, these include: Age, Gender, Smoking_History, Tumor_Size_mm, Tumor_Location, Stage, Treatment, Survival_Months, Ethnicity, Insurance_Type, Family_History, Comorbidity_Diabetes, Comorbidity_Hypertension, Comorbidity_Heart_Disease, Comorbidity_Chronic_Lung_Disease, Comorbidity_Kidney_Disease, Comorbidity_Autoimmune_Disease, Comorbidity_Other, Performance_Status, Blood_Pressure_Systolic, Blood_Pressure_Diastolic, Blood_Pressure_Pulse, Hemoglobin_Level, White_Blood_Cell_Count, Platelet_Count, Albumin_Level, Alkaline_Phosphatase_Level, Alanine_Aminotransferase_Level, Aspartate_Aminotransferase_Level, Creatinine_Level, LDH_Level, Calcium_Level, Phosphorus_Level, Glucose_Level, Potassium_Level, Sodium_Level, Smoking_Pack_Years. We will now take a look at how these are distributed among the cancer stages, shown in picture 1.



Picture 1 Distribution of features among cancer stages

We can see that the data is very equally distributed among the classes. Besides that, the data has a very low correlation with each other, it is around 0. In the final set of graphs we can see that the data has a rather uniform distribution, which means it is most likely synthetic in origin.

3. Related work

There are 8 code submissions on Kaggle for this data set. 5 of them are an analysis of the dataset, and they have come to similar conclusions to us. The data set is very balanced, each of the categorical labels are exactly equal, for example all of the Stage categories take up

~25%, there is almost an exactly 50/50 split between men and women, there is 20% of each ethnicity and so on.
Other 3 code submissions are attempts at solving the problem. They involve encoding the categorical columns, dropping irrelevant columns and other preprocessing techniques. They used a variety of classifiers and composition classifiers such as AdaBoost, CatBoost, LightGBM, Perceptron, Ridge, Random Forest, Decision Tree and so on… all of them got a macro F1 score result around 0.25

4. Methodology

We have approached the problem in a multitude of ways.We will describe them separately

Approach 1:

Approach 1 included engineering features, specifically merging Comorbidity columns into a single Comorbidity_Count column, as well as Label Encoding other categorical columns. A number of columns were dropped since it was determined for them not to have a good impact on the final result, these were: Hemoglobim_Level,White_Blood_Cell_Count, Platelet_Count, Aspartate_Aminotransferase_Level, Creatinine_Level, LDH_Level, Calcium_Level, Insurance_Type, as well as the Patient_ID. SimpleImputer has been used to add missing data, but I'm not even sure there was any missing data. StandardScaler was used to scale the remaining columns.

Approach 2:

A Grid Search with 5-fold cross-validation was conducted to optimize hyperparameters of Random Forest Classificators. The best parameters found were: max_depth=20, min_samples_leaf=2, min_samples_split=10, and n_estimators=100. The model achieved a macro F1 score of 0.2563 on the test set. Feature importance analysis and confusion matrix were generated for further insights.

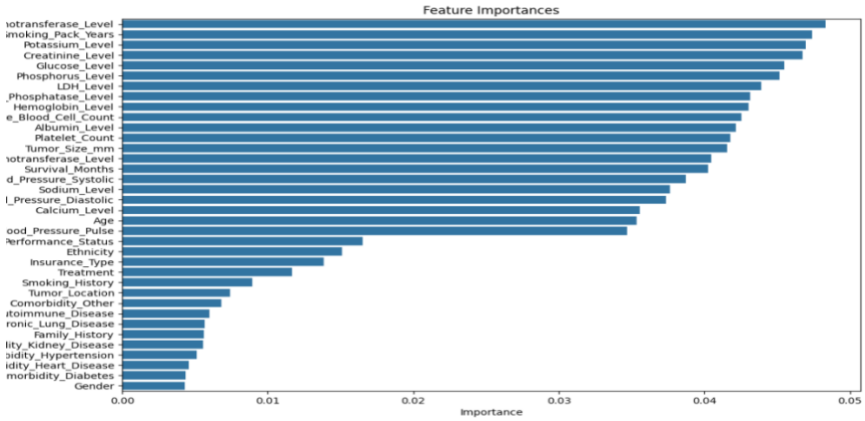
Approach 3:

A VotingClassifier used in project combines Decision Tree, KNN, and SVM models, each optimized via Grid Search. Using soft voting, it predicts class probabilities and averages them, enhancing overall performance.

Approach 4:

This approach includes the following preprocessing steps:

- Dropping some of the columns that are deemed to be of the least importance by ExtraTreesClassifier, as is demonstrated in picture 2, including Patient_ID column



Picture 2 Feature Importances

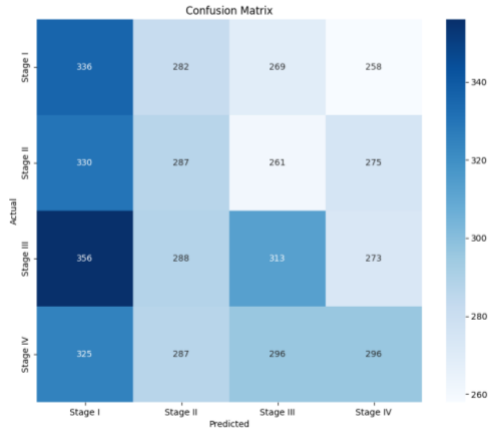
- Using KNN Imputer, Standard Scaler and PCA
- Trying to classifying data with different classifiers. Ensembles tried:
  - ❖ HistGradientBoostingClassifier
  - ❖ AdaBoostClassifier with optimized Decision Tree
  - ❖ ExtraTreesClassifier

5. Discussion

Again as with the last chapter, we will discuss the approaches separately.

Approach 1:

The data set has been split 80:20 into a train set and a test set. We have used the train set with a SVC classifier which had the following hyperparameters: kernel='rbf', gamma='scale', C=5. Rbf was used as we have found the data set to be rather non-linear. Other two were found with trial and error by hand. Final F1 Macro score was 0.2601 which is around the 0.25 others have been getting. I will blame the data set for this low score as it is very synthetic and possibly even random generated.



	precision	recall	f1-score	support
0	0.25	0.29	0.27	1145
1	0.25	0.25	0.25	1153
2	0.27	0.25	0.26	1230
3	0.27	0.25	0.26	1204
accuracy			0.26	4732
macro avg	0.26	0.26	0.26	4732
weighted avg	0.26	0.26	0.26	4732

Macro F1 Score: 0.26013054927223056

Picture 3 SVM Confusion matrix and Classification report

We can see that the model heavily favours Stage 1, and is a little bit more precise with Stage 4.

Approach 2:

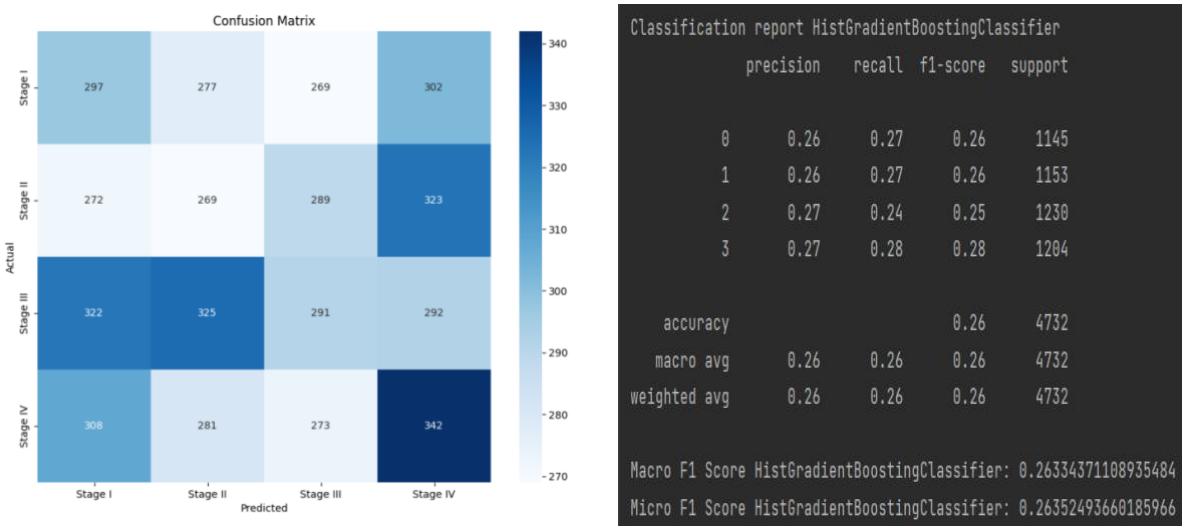
Next classification attempt was performed using Decision Tree, KNN, and SVM classifiers, optimized through Grid Search. Data preprocessing included encoding categorical features and standardizing numerical features. Grid Search with 5-fold cross-validation determined optimal hyperparameters: Decision Tree (max_depth=30, min_samples_leaf=1, min_samples_split=5), KNN (n_neighbors=10, weights=distance), and SVM (C=5, gamma=scale, kernel=rbf). These models were combined in a VotingClassifier with soft voting. The ensemble achieved a macro F1 score of 0.2492 on the test set.

Approach 3:

This ensemble approach leverages the strengths of individual classifiers, achieving a macro F1 score of 0.25 on the test set.

Approach 4:

Hyperparameters for Imputer, Scaler and PCA were optimized manually. As for the classifiers, they were mostly optimized by the Grid Search algorithm. The hyperparameters that were the hardest to tune were the learning rate and random state, which was expected due to their nature and purpose, and they were optimized manually. Precision, recall and F1 scores for each individual class by each classifier can be found in the classification reports written in the terminal. The highest achieved F1 score was 0.265, thanks to Hist Gradient Boosting Classifier. Picture 4 shows its confusion matrix as well as the classification report. The model demonstrates a clear bias towards Stage 4, which could be explained by the fact that data was possibly randomly generated.



Picture 4 Hist Gradient Boosting Classifier Confusion matrix and Classification report

6. References

Data set: https://www.kaggle.com/datasets/rashadrmammadov/lung-cancer-prediction/data

Code submissions: https://www.kaggle.com/datasets/rashadrmammadov/lung-cancer-prediction/code