Pancreatic Cancer: Diagnosis Classification Model

Ben Gronbach

School of Engineering and Computer Science

Bioengineering Department

University of the Pacific

Stockton, California

January 17

Abstract:

Pancreatic cancer has a 5-year survival rate of less than 10%, classifying it as one of the deadly forms of cancer in the United States. When detected at an early stage, viable treatments are considerably less invasive and significantly more effective, improving patient survival rate as well as quality of life. However, early detection remains a major clinical challenge due to the nature of the disease. Pancreatic cancer often produces nonspecific symptoms, or no symptoms whatsoever, therefore frequently going undiagnosed until the disease has progressed beyond curative stages.

In this study, we trained a machine learning classification model on urinary biomarker data from three groups of patients; Healthy controls, patients with non-cancerous pancreatic conditions, and cancer patients. The data is from a study published in *PLOS medicine* in 2020, by Silvana Debernardi and colleagues. The data contains 590 samples of patients with varying ages and sexes. It contains urinary data which tracks six biomarkers; CA 19-9 Plasma, creatinine, LYVE1, REG1B, TFF1, and REG1A. These biomarkers have been shown to exhibit differential expression across disease states, making them promising candidates for noninvasive pancreatic cancer screening.

Model Background

For this purpose, we decided to utilize a XGBoost Classifier model. Ensemble models are effective for handling classification problems due to their ability to handle nonlinear functions, and in this particular case we needed a model that was adept at handling missing data. XGBoost was a logical choice considering the medium size of this dataset, and its ability to improve its performance over the multiple iterations, allowing for accurate classification of complex patient cases. Biomarker data often has various magnitude concentrations, which XGBoost handles well by focusing on the relationship between features rather than the magnitude or scale of their values.

Model Validation and Accuracy Metrics

To track the accuracy of this model, we utilized a variety of insightful metrics. First, Receiver Operating Characteristic – Area under the curve (ROC-AUC) was utilized, which is commonly used in classification models. It represents the probability that when given two random examples, the model will weigh a positive example over a negative one. In this case, it represents the percentage chance that the model will designate a higher cancer probability to a cancer patient over a healthy control.

Additionally, a confusion matrix was utilized, to understand where the model was getting confused on the test set. It lays out the models predictions against the actual classes in a clear and interpretable manner.

Next, we tracked recall sensitivity. This was used to measure the percentage of cancer cases the model caught. Precision is another similar metric, which in this case was tracking how many cancer cases were reported correctly compared to incorrectly. In other words, the amount of false positives the model reported.

Python Code

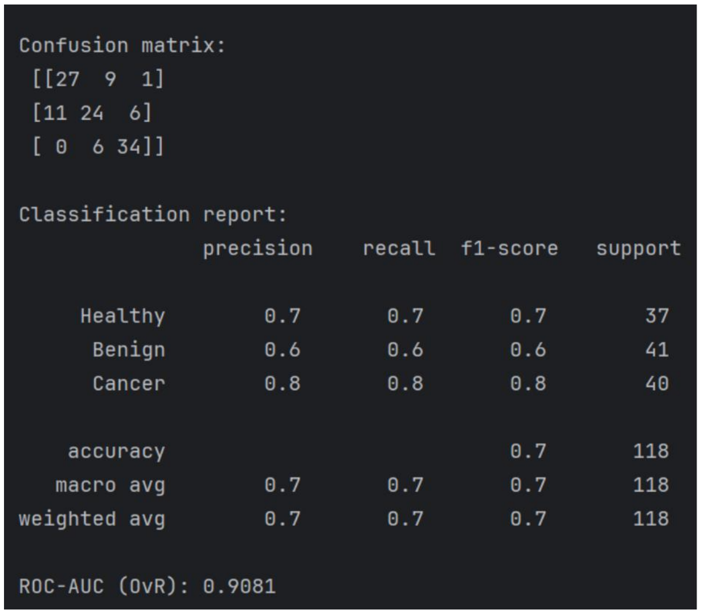
```
import os

import numpy as np

import kagglehub

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

from xgboost import XGBClassifier


# Load Dataset

path = kagglehub.dataset_download("johnjdavisiv/urinary-biomarkers-for-pancreatic-cancer")

print("Path to dataset files:", path)

print("Files:", os.listdir(path))


csv_path=os.path.join(path,'Debernardi et al 2020 data.csv')

df=pd.read_csv(csv_path)


# Drop Columns

df=df.drop(columns=['sample_id','patient_cohort','sample_origin','stage','benign_sample_diagnosis'])


# Convert sex into numerical data
```

```
df["sex"] = df["sex"].map({"M":1, "F":0})


# Define Target


target_column='diagnosis'

X=df.drop(target_column,axis=1)

y=df[target_column]

y=y-1

class_names=["Healthy", "Benign", "Cancer"]


#Split Data


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)


# Build Model


model=XGBClassifier(

   n_estimators=500,

   learning_rate=0.05,

   max_depth=5,

   subsample=0.8,

   colsample_bytree=0.8,

   reg_lambda=1.0,

   reg_alpha=0.0,

   min_child_weight=1,

   gamma=0.0,

   random_state=42,

   n_jobs=-1

)


# Increase Weight of Cancer Mistakes
```

```
cancer_class=2

cancer_weight=90

sample_weight=np.where(y_train == cancer_class, cancer_weight, 1.0)


# Train Model


model.fit(X_train, y_train,sample_weight=sample_weight)


#Lower Threshold for Cancer Detection


cancer_threshold=0.2

y_proba = model.predict_proba(X_test)

y_pred_adjusted = np.argmax(y_proba,axis=1)

cancer_prob=y_proba[:,cancer_class]

y_pred_adjusted[cancer_prob >= cancer_threshold]=cancer_class


# Evaluate Model Accuracy

print("\nConfusion matrix:\n", confusion_matrix(y_test, y_pred_adjusted))

print("\nClassification report:\n", classification_report(

  y_test,

  y_pred_adjusted,

  target_names=class_names,

  digits=1

)

    )


#ROC-AUC Metric




roc_auc=roc_auc_score(y_test, y_proba,multi_class='ovr')
```

```
print(f"ROC-AUC (OvR): {roc_auc:.4f}")
```

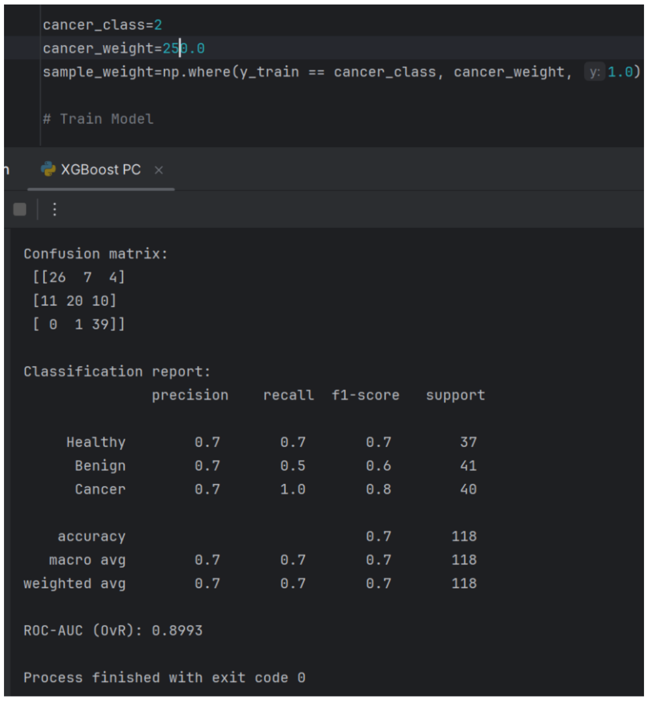
Results and Discussion



```
Confusion matrix:
 [[27  9  1]
 [11 24  6]
 [ 0  6 34]]

Classification report:
              precision    recall  f1-score   support

     Healthy        0.7       0.7       0.7        37
      Benign        0.6       0.6       0.6        41
      Cancer        0.8       0.8       0.8        40

    accuracy                            0.7       118
   macro avg        0.7       0.7       0.7       118
weighted avg        0.7       0.7       0.7       118

ROC-AUC (OvR): 0.9081
```

Figure 1. Initial Accuracy Results

Initially after running the model, and some basic hyperparameter tuning, our model scored an ROC-AUC score of 0.9081, showing it had a good sense of accuracy when deciphering between classes. It was able to correctly classify 70% of healthy patients and 80% of cancer patients, which is a relatively high accuracy. It struggled more when dealing with benign patients, suggesting some overlap in Biomarker expression between the benign class and the other two. While correctly identifying 34 of 40 subjects in the cancer class may be considered a high accuracy, in this case the primary goal is shifted. Seeing as a false negative has much worse ramifications compared to a false positive. Therefore, the goal in the case of this model is to correctly diagnose all cancer patients correctly, even if that may mean sacrificing some overall accuracy and diagnosing more false positives. To accomplish this, we increased the weight of the cancer class, increasing punishment for when the model mislabeled a cancer patient. However, to achieve high recall, the weight had to be increased dramatically, to 250x the original weight (see figure 2 below).



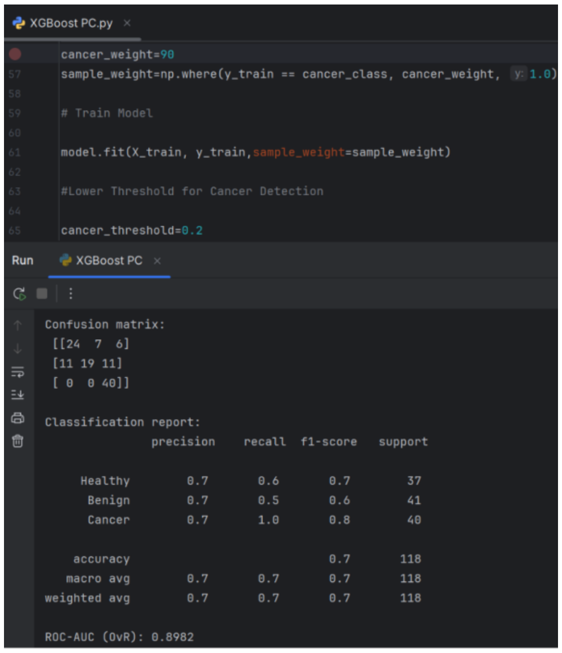
```
cancer_class=2
cancer_weight=250.0
sample_weight=np.where(y_train == cancer_class, cancer_weight, y: 1.0)

# Train Model
```

XGBoost PC

```
Confusion matrix:
 [[26  7  4]
 [11 20 10]
 [ 0  1 39]]

Classification report:
               precision    recall  f1-score   support

     Healthy       0.7       0.7       0.7        37
      Benign       0.7       0.5       0.6        41
      Cancer       0.7       1.0       0.8        40

    accuracy                           0.7       118
   macro avg       0.7       0.7       0.7       118
weighted avg       0.7       0.7       0.7       118

ROC-AUC (OvR): 0.8993

Process finished with exit code 0
```

Figure 2. Accuracy results when dramatically increasing cancer weights.

This resulted in a much higher cancer recall, with the model correctly identifying 39/40 cancer cases. This lowered overall accuracy, with the ROC-AUC value dipping to 0.8993. However, this practice is unsustainable as it often leads to a model learning to only predict cancer due to the high reinforcement weight put on it. The model can potentially overfit to cancer and lose any pattern recognition it had. To combat this, we lowered the cancer weight, while also lowering the threshold for cancer classification. This tells the model cancer is important, while still forcing it to learn underlying patterns within the data. Decreasing the threshold means that the model will designate more patients as cancerous, by pushing it to err on the side of caution when it is unsure. The cancer weight was lowered from 250 to 90, a moderately high value, while the cancer threshold was lowered from 0.50 to 0.20 (see figure 3 below).


XGBoost PC.py

```
cancer_weight=90
sample_weight=np.where(y_train == cancer_class, cancer_weight, y: 1.0)

# Train Model

model.fit(X_train, y_train,sample_weight=sample_weight)

#Lower Threshold for Cancer Detection

cancer_threshold=0.2
```

Run XGBoost PC

```
Confusion matrix:
 [[24  7  6]
 [11 19 11]
 [ 0  0 40]]

Classification report:
              precision    recall  f1-score   support

     Healthy       0.7       0.6       0.7        37
      Benign       0.7       0.5       0.6        41
      Cancer       0.7       1.0       0.8        40

    accuracy                           0.7       118
   macro avg       0.7       0.7       0.7       118
weighted avg       0.7       0.7       0.7       118

ROC-AUC (OvR): 0.8982
```

Figure 3. Accuracy results with moderate cancer weight and lowered threshold

This iteration of the model scored a lower overall accuracy, with an ROC-AUC of 0.8982. It correctly identified 40/40 cancer patients, while reporting 17 false negatives. This represents the tradeoffs when configuring a model to predict cancerous results. A clinical decision must be made when deciding between a higher accuracy and higher recall model. It is obviously the intention to catch all cancer cases, but if that leads to an inaccurate model that is something that must be taken into account and balanced.

Conclusion

Through this project, I gained valuable experience in machine learning engineering within a medical context. I learned how to prioritize certain outcomes within the model and improved my hyperparameter tuning skills. I also learned about the important concept of balancing two different goals, by measuring the tradeoff between model recall and accuracy. Working with a medical dataset and achieving a high degree of accuracy was enjoyable and meaningul, and further excites me for the future of machine learning and artificial intelligence in the healthcare sector.

If I were to take this project further, I would run a cost benefit analysis between the outcomes of a false negative and false positive to accurately balance the model accordingly. By understanding the financial ramifications of both cases, it would further inform my decision on which values to prioritize in the model.

Thank you for taking the time to read this report, I am open to any suggestions, questions, and critiques that you may have, so please reach out to me.

References

Davis, J. (2020, December 10). *Urinary biomarkers for pancreatic cancer*. Kaggle. https://www.kaggle.com/datasets/johnjdavisiv/urinary-biomarkers-for-pancreatic-cancer