ML-PETIL: A Machine Learning Predictor of the Expansion of Tumor Infiltrating Lymphocytes in Patients' Bladder Tumors

Kayode D. Olumoyin
Integrated Mathematical Oncology Department
H. Lee Moffitt Cancer Center and Research Institute
Tampa, FL 33612

Ahmet M. Aydin
Department of Urology
University of Arkansas for Medical Sciences
Little Rock, AR 72205

Brittany L. Bunch
Iovance Biotherapeutics, Inc.
San Carlos CA

Shari Pilon-Thomas
Department of Immunology & Genitourinary Oncology
H. Lee Moffitt Cancer Center and Research Institute
Tampa, FL 33612

Michael A. Poch
Department of Genitourinary Oncology
H. Lee Moffitt Cancer Center and Research Institute
Tampa, FL 33612

Katarzyna A. Rejniak
Integrated Mathematical Oncology Department
H. Lee Moffitt Cancer Center and Research Institute
Tampa, FL 33612

Abstract

It is estimated that over 720,000 people are living with bladder cancer in the United States. Many of these patients could benefit from immune-based therapies, such as adoptive T cell therapy (ACT) that uses patients' autologous tumor-infiltrating T lymphocytes (TILs). To determine whether a patient will be a good candidate for this immunotherapy, we developed a mathematical model to predict whether the tumor isolated from a patient will grow T cells. In this work, we present a machine learning protocol that uses retrospectively collected patient tumor data with either demographic, clinical, or biological tumor sample features. The proposed protocol consists of a suite of machine learning algorithms that identifies a robust set of predictive features which are used to discriminate the dataset into two classes, those with successful TIL expansion or with no TILs. Our machine learning protocol optimize for both the classifier type and the hyperparameters specific to each classifier. We demonstrate performance of this protocol using metrics that include precision, specificity, classification accuracy, and sensitivity.

Keywords Tumor Infiltrating Lymphocytes (TILs) · Adoptive Cell Therapy (ACT) · Feature Selection · Random Forest · Support Vector Machines · Hyperparameters

1 Introduction

It has long been known that Bladder cancer is a malignancy that is responsive to immune-based therapy (Morales et al., 1976; Poch et al., 2018; Aydin et al., 2021). In the early 1970s, patients with localized non-muscle invasive bladder cancer (NMIBC) were treated with intravesical installations of Bacillus Calmette-Guerin (BCG) (Morales et al., 1976). This treatment option for Bladder cancer patients have demonstrated a 20% response rate overall (Powles et al., 2014). Unfortunately, the median overall survival for patients who received this treatment is worse for those with metastatic disease (Poch et al., 2018). Previous studies have identified tumor-infiltrating lymphocytes (TIL) within bladder cancer specimens. The presence of TIL is associated with improved survival in patients with muscle-invasive urothelial carcinoma. One mechanism of harnessing the power of TIL is through adoptive cell transfer (ACT). This is composed of the extraction of TIL from human tumor samples, ex vivo expansion, and reinfusion of expanded autologous lymphocytes into patients (Rosenberg and Restifo, 2015). ACT using tumor-infiltrating lymphocytes (TIL) has shown promise for patients with metastatic melanoma, and cervical cancer (Besser et al., 2010; Pilon-Thomas et al., 2012).

In (Poch et al., 2018), the feasibility of TIL expansion has been shown to be about 70% of collected primary bladder tumors. This process takes weeks, so it will be beneficial for clinicians to have access to a tool early that can predict whether a patient will benefit from TIL therapy using demographic data, clinical data, and biological tumor sample (BTS) data. Machine learning has found application in many scientific domains such as developing and updating mathematical models that describes the evolution and treatment of diseases including bladder cancer (Zhang et al., 2017; Garapati et al., 2017; Mi et al., 2021; Przedborski et al., 2021). In (Zhang et al., 2017), Support Vector Machine (SVM) classifiers were used to describe the textural features of bladder cancer from diffusion-weighted images. In (Garapati et al., 2017), machine learning classifiers including Linear Discriminant Analysis (LDA), Neural Network (NN), Support Vector Machine (SVM), and Random Forest (RF) algorithm were used to evaluate the feasibility of assessing bladder cancer stage in CT urography (CTU). In (Tokuyama et al., 2022), the authors developed a machine learning approach that predicts the early recurrence of non-muscle invasive bladder cancer (NMIBC) based on morphological features extracted from hematoxylin and eosin (H&E) stained slide images. In (Mi et al., 2021), a predictive model for the response to chemotherapy in muscle-invasive bladder cancer (MIBC) was developed using H&E images – features were extracted from these images and with predictions obtained from the predictive model approach, they were able to stratify patients into good and poor survival group.

2 Methodology

In Figure 1, we propose a machine-learning predictor of the expansion of tumor-infiltrating lymphocytes (ML-PETIL) using machine-learning classifiers such as Support Vector Machine (SVM) classifiers (Vapnik, 1995; Guyon et al., 2002; Chapelle et al., 1999). SVM classifiers are particularly well suited for classification of small dataset (Géron, 2019). In this paper, we refer to features to mean the attributes collected about patients' and about patients' tumors. These includes demographic, clinical, and biological tumor sample (BTS) features. We employ feature selection algorithms such as the Random Forest (RF) algorithm (Breiman, 2001, 1984), and the minimal-redundancy-maximal-relevance (mRMR) algorithm (Peng and Ding, 2005; Ding and Peng, 2005) to obtain a ranking of the features in our dataset. The RF algorithm generates feature importance score (FIS) for each feature. The features with high FIS are considered more important than those with lower FIS. FIS is affected by the fidelity of the dataset, imbalance in the dataset and missing dataitems in the dataset. In the case of mRMR, mutually independent features are considered to be more important than those that are highly correlated (Peng and Ding, 2005; Ding and Peng, 2005). On the other hand, Principal Component Analysis (PCA) creates a dimensionality reduction of the feature space (Washington et al., 2020). PCA minimizes the influence of noise or imperfect data by identifying the principal components in a feature space. Using ensemble learning methods (Tan and Gilbert, 2003), we are able to combine the feature selection algorithms and set a threshold that helps us select the most robust features. These selected features are used to train a machine learning classifier such as SVM classifiers.

In the machine learning protocol (ML-PETIL) presented in Figure 1, there are classifier-specific hyperparameters which we learn in a data-informed approach. For instance, each of the many different SVM classifiers have a set hyperparameters. There is a parameter space of these hyperparameters, where each point in this space is a set of hyperparameters that corresponds to a classifier. We need to efficiently search within this parameter space for sets of hyperparameters that corresponds to optimal classifiers. In the implementation of ML-PETIL, we have used the function `'GridsearchCV'` to search for the set of hyperparameters that produces the most optimal classifier for our dataset. `'GridsearchCV'` is a Python-based function available in Scikit-Learn, a free software machine learning library (Abraham et al., 2014).

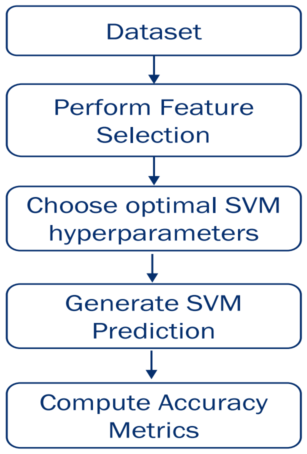


Figure 1: ML-PETIL (Machine Learning Predictor of the Expansion of Tumor Infiltrating Lymphocytes)

2.1 Data

The dataset for ML-PETIL consists of features that are either numerical (continuous or discrete) or nominal. These features represent attributes collected about patients and about patients' tumors. Features could be of the following data types:

- Numerical features:
  1. Positive real-valued data, which takes values in the real line, i.e., (BMI, Tumor size (cm), CD3 % of live (tumor digest), . . .)
  2. (Discrete) count data, which takes values in the natural number, i.e., (Age at Surgery, Tumor digest count-primary tumor, Number of fragments plated)
- Nominal features:
  1. Categorical data, which takes values in a finite unordered set, i.e., Molecular subtype, which could be (Lumil, Basal, Double Negative), Ethnicity, which is one of (Hispanic, Not Hispanic, Unknown), Surgery (Radical Cystectomy, Trans urethral resection of bladder tumor (TURBT)).

2. Ordinal data, which takes values in a finite ordered set, e.g., Smoker which takes values from (Never, ever, current).

The nominal features are assigned natural numbers specifying the category each feature data point belongs. The machine learning algorithms in ML-PETIL requires that we perform normalization of each feature except the Random Forest algorithm. Each feature was normalized separately either by the z-score, i.e., each feature is centered at 0 or by standardization between 0 and 1. We note that normalization is done before we split the dataset into training and testing cohorts. While this ensures that the training and testing set share similar distribution, it has been reported that this approach could leak part of the testing set into the training set, we avoid this by using 'Pipeline', a functionality that is available in Scikit-Learn (Géron, 2019).

2.2 Feature Selection

In ML-PETIL, we use 3 feature selection algorithms: Random Forest (RF), minimal-redundancy-maximal-relevance criterion (mRMR), and Principal Component Analysis (PCA). In this section, we discuss the Random Forest algorithm and how it generates feature importance scores (FIS) for all the features in our dataset. Random Forest (RF) is an ensemble learning of decision trees. To understand how RF algorithm generates FIS for each feature, we start by describing 'decision tree' and how to obtain FIS from it using a small classification dataset (Table 1). We also describe an ensemble learning for classification called bootstrap aggregating (bagging) (Breiman, 1984) commonly used to aggregate the FIS from several decision trees in a Random Forest.

Patients	Surgeon	Age at Surgery	BMI	Target
patient 01	Surgeon D	68	21.05	No TIL
patient 02	Surgeon D	57	33.76	Grew TIL
patient 03	Surgeon A	61	26.28	Grew TIL
patient 04	Surgeon A	91	19.22	Grew TIL
patient 05	Surgeon B	79	26.71	Grew TIL
patient 06	Surgeon A	71	20.19	No TIL
patient 07	Surgeon D	73	33.95	No TIL
patient 08	Surgeon B	78	26.33	No TIL
patient 09	Surgeon D	68	21.32	Grew TIL
patient 10	Surgeon D	69	34.21	Grew TIL

Table 1: A 10 patients dataset with 3 features and 1 target comprised of 2 distinct classes

A decision tree (Quinlan, 1993) is comprised of nodes connected by edges. There is a starting node (parent node), decision node (a child node that may generate other nodes based on a decision threshold), and the leaf node (a terminal node). To build a decision tree from the dataset in Table 1, we replace the categorical feature 'Surgeon' with values assigned as follows: `Surgeon A = 1, Surgeon B = 2, Surgeon D = 4`. Next, we perform random sub-sampling, where we select 7 out of the 10 patients in this dataset. We build a decision tree on this sub-sample dataset using `'DecisionTreeClassifier'` a function available in Scikit-Learn. The tree generated by `'DecisionTreeClassifier'` can be visualized using the function `'export_graphviz()'` (also available in Scikit-learn). The function `'export_graphviz()'` creates a `'.dot'` file – a file we convert to a `'.png'` file from the command line, using: `'dot -Tpng filename.dot -o filename.png'`.

At each non-leaf node, `'DecisionTreeClassifier'` implements the classification and regression tree (CART) algorithm (Breiman, 1984), which Identifies a starting feature and a splitting threshold. The CART algorithm implemented in Scikit-Learn produces binary trees, where non-leaf nodes always have two children. CART is a greedy algorithm: it greedily searches for a split at the parent node and then repeats this process at each subsequent non-leaf nodes. Greedy algorithms often produce good solutions that are not necessarily optimal (finding an optimal tree is an NP-Complete problem (Géron, 2019)). Each non-leaf node generated by `'DecisionTreeClassifier'` has five attributes (a leaf node has four attributes): (1) a splitting threshold (not available in a leaf node), (2) a `criterion` attribute, (3) a `sample` attribute, (4) a `value` attribute, and (5) a `class` attribute. CART algorithm chooses a feature at each non-leaf node, and it identifies a splitting threshold.

The criterion attribute in `'DecisionTreeClassifier'` is usually the `gini impurity` (Nembrini, 2018; Menze et al., 2009) or `entropy` (a thermodynamics concept which later found application in Shannon's information theory) (Shannon, 1997). In machine learning, we need a way to measure the quality of the splitting threshold specified by CART algorithm at each non-leaf node, in our case we use either `gini impurity` or `entropy` (Murphy, 2012). `Entropy` tends to produce slightly more balanced trees, but they take longer to train (Géron, 2019). The `gini impurity` G_i at the ith node is computed as:

$$G_i = 1 - \sum_{j=1}^{n} \tag{2.2.1}$$

In (2.2.1), n is the number of distinct classes, in our case $n = 2$ and $p_{i,j}$ which is the probability that a training instance at the ith node belongs to class j is computed as the ratio of class j instances among the training instances at the ith node. The maximum `gini impurity` is 0.5, this occurs when the two distinct classes have the same number of training instances at that node. The minimum `Gini impurity` is 0, which means that all the training instances at that node belongs to the same class. The `Sample` attribute of a node denotes the number of training instances at that node. `Value` attribute of a node tell you how many instances of each class at that node. The `class` attribute of a node is the predicted class for the training instances at that node which satisfies the splitting threshold. The sub-sample of 7 patients randomly selected from 10 patients in Table 1 are `'patient 01'`, `'patient 02'`, `'patient 03'`, `'patient 04'`, `'patient 06'`, `'patient 07'`, and `'patient 08'`. The corresponding decision tree generated using this sub-sample is shown in Figure 2. The orange node is the parent node `(node 0)`, its `gini impurity` is computed as $G_0 = 1 - (4/7)^2 - (3/7)^2 = 0.49$. The decision nodes are in purple. The `gini impurity` of the decision node `(node 2)` is $G_2 = 1 - (2/5)^2 - (3/5)^2 = 0.48$. The `gini impurity` of the decision node `(node 3)` is $G_3 = 1 - (2/3)^2 - (1/3)^2 = 0.444$. The leaf nodes are shown in green.

Next, we compute the `gini impurity decrease` (Nembrini, 2018) at each non-leaf node in Figure 2.

$$Gini\ Impurity\ Decrease = \frac{N_t}{N}\left(G_i - \frac{N_{t,R}}{N_t}G_{i,R} - \frac{N_{t,L}}{N_t}G_{i,L}\right) \tag{2.2.2}$$

In (2.2.2), N is the total number of samples in the training data, N_t is the number samples at node i, $N_{t,R}$ is the number of samples in the right child of node i, $N_{t,L}$ is the number of samples in the left child of node i, $G_{i,R}$ is the `gini impurity` in the right child of node i, and $G_{i,L}$ is the `gini impurity` in the left child of node i.

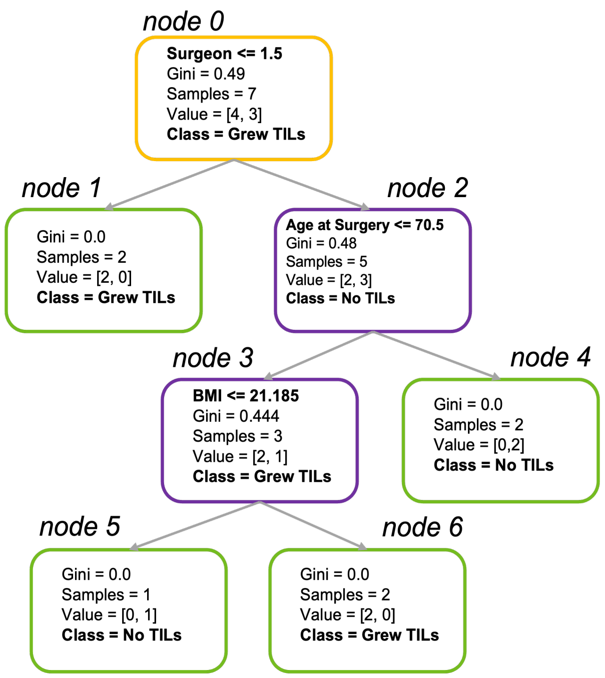


Figure 2: **A decision tree showing decision thresholds and node predictions at each non-leaf node, using a sub-sample of 7 of the 10 patients' data in Table 1**

$$Gini\ Impurity\ Decrease\ (node0) = \frac{7}{7}\left(0.49 - \frac{5}{7}(0.48) - \frac{2}{7}(0)\right) = 0.147143$$

$$Gini\ Impurity\ Decrease\ (node2) = \frac{5}{7}\left(0.48 - \frac{2}{5}(0) - \frac{3}{5}(0.444)\right) = 0.152571$$

$$Gini\ Impurity\ Decrease\ (node3) = \frac{3}{7}\left(0.444 - \frac{2}{3}(0) - \frac{1}{3}(0)\right) = 0.190286$$

The `gini impurity decrease` at `node 0`, `node 2`, `node 3` is for the features '`Surgeon`', '`Age at Surgery`', and '`BMI`' respectively. The feature importance score (FIS) (Menze et al., 2009) is computed according to the following formula:

$$FIS_{particular feature} = \frac{\sum\ of\ impurity\ decrease\ corresponding\ to\ a\ particular\ feature}{\sum\ of\ all\ gini\ impurity\ decrease} \tag{2.2.3}$$

We compute FIS for the features '`Surgeon`', '`Age at surgery`', and '`BMI`' denoted as $FIS_{Surgeon}$, $FIS_{AgeatSurgery}$, and FIS_{BMI} respectively.

$$FIS_{Surgeon} = \frac{0.147143}{0.147143 + 0.152571 + 0.190286} = 0.300292$$
$$FIS_{AgeatSurgery} = \frac{0.152571}{0.147143 + 0.152571 + 0.190286} = 0.311369$$
$$FIS_{BMI} = \frac{0.190286}{0.147143 + 0.152571 + 0.190286} = 0.388339$$

FIS depend on the data the decision tree is trained on. For instance, when we swap `'patient 04'` for `'patient 05'` in the training data (Table 1), we obtain a decision tree different from the one presented in Figure 2. We obtain the following FIS for the features `'Surgeon'`, `'Age at surgery'`, and `'BMI'`:

$$FIS_{Surgeon} = 0,\ FIS_{AgeatSurgery} = 0.5625,\ FIS_{BMI} = 0.4375.$$

If we build multiple decision trees from sub-samples of the dataset in Table 1, the FIS for each of the above 3 features will have high variance. One way to solve the high variance problem is to use Random Forest (Murphy, 2012). In the original Random Forest algorithm developed by Leo Breiman, he used Bootstrap aggregating (bagging) – an ensemble learning method mostly used when we have a noisy dataset (Breiman, 1984, 2001). In bagging, several random sub-samples of the training data are selected with replacement. Each sub-sample is used to build a decision tree and for each feature, we compute a statistical mode of all FIS obtained from all the decision trees. Another ensemble method used in the Random Forest algorithm is pasting. In pasting, the random sub-samples of the training data are selected without replacement. We implement Random Forest algorithm by calling `'RandomForestClassifier()'` a function available in Scikit-Learn. The following parameters of `'RandomForestClassifier()'`: `'n_estimators=integer'`, `'criterion=[gini, entropy]'`, `'max_samples=float'`, and `'bootstrap=bool'` can be used to choose the ensemble method we want to implement. For instance, when we set `'bootstrap'` to True, `'RandomForestClassifier()'` implements bagging ensemble method. In which case, we need to set the parameter `'max_samples'` a float value. `'Max_samples'` denotes the proportion of the training data we want in each sub-sample that generates the decision trees. We use `'n_estimators'` to specify the number of decision trees in the ensemble. For the dataset in Table 1, when we implement `'RandomForestClassifier(n_estimators=500,criterion=gini, max_samples=0.7, bootstrap=True, ...)`, we obtain:

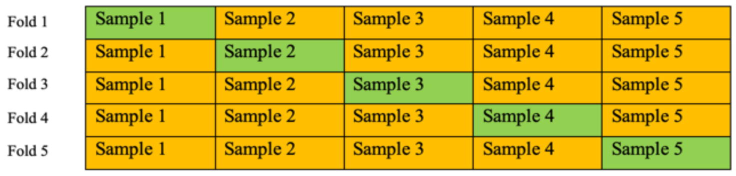
$$FIS_{Surgeon} = 0.1479,\ FIS_{AgeatSurgery} = 0.3975,\ FIS_{BMI} = 0.4546.$$

2.3 K-fold Cross Validation

We use k-fold cross validation when training the classifier in ML-PETIL on the training data. The number of k-fold that generates the highest average validation accuracy for the training set is also an hyperparameter we learn from the data in ML-PETIL. The use of cross validation techniques during training prevents overfitting and helps us to build a more generalized machine learning model. k-fold cross validation is an effective way to measure how well a machine learning algorithm performs on the training set. The training set is divided into k non-overlapped subsets with an equal amount of data. Suppose we choose $k = 5$, (Table 1) then for each fold, one of the 5 subsets is chosen as the validation set ('green'), and the other $k - 1$ subsets ('yellow') are combined to form the training set. validation score is obtained in each fold, and the average validation score is obtained from the validation score of each fold (Xu et al., 2017).

2.4 Machine Learning Classifier

In ML-PETIL, we optimize for the choice of a classifier by using a data-informed approach. First, we split our dataset into training and testing cohort in a $70/30$ split. Each classifier has a set of hyperparameters which we learn from the training data. We prioritize the classifier that produces the most optimal nonlinear higher dimensional hyperplane which discriminates the training data. We utilize k-fold cross-validation on the training data, which improves the generalization of the classifier to the dataset. Using a k-fold cross validation also reduces the chance of overfitting to the training data.



Fold 1	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Fold 2	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Fold 3	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Fold 4	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Fold 5	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5

Figure 3: **A five-fold cross validation**. Each row represents a fold of the training data sub-divided into 5 samples. In each fold, we train the classifier on four samples (in orange) and validate on one sample (in green).

We briefly discuss how to construct support vector machines (SVMs). SVM classifiers are a type of supervised learning algorithm that can be used for classification tasks in cancer biology (Guyon et al., 2002; Abreu et al., 2016). They work by optimizing the hyperplane that discriminates a dataset into different classes usually by transforming the training data into a nonlinear higher-dimensional space. The fundamental idea in the SVM algorithm is finding the decision boundary that stays far away from the training data, while optimally separating the training data into distinct classes. This concept is called large margin classification. A classifier with a large margin will generalize better than a classifier whose decision boundary is close to the training data when tested on new data. When the decision boundary that discriminate a dataset is linear, we say the dataset is linearly separable. However, most dataset are not linearly separable. In this section, we present how to construct a linear SVM – an example of a large margin classifier whose decision boundary is linear. We show how to extend linear SVM to nonlinear SVM – a large margin classifier whose decision boundary is nonlinear.

For simplicity, consider a binary classification task, where the dataset $\mathbf{x}$ consists of M data points and P features. We define $\mathbf{x}$ as follows:

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}^1 \\ \mathbf{x}^2 \\ \vdots \\ \mathbf{x}^M \end{pmatrix} \tag{2.4.1}$$

Where each data point $\mathbf{x}^i$ in (2.4.1) is given by $\mathbf{x}^i = \left(x_1^i, x_2^i, \ldots, x_P^i\right)$ for $i \in \{1, 2, \ldots, M\}$. And for each i, the corresponding target is the binary class $y^i \in \{-1, 1\}$. A Linear SVM classifier generates a predicted binary class $\hat{y}^i$ for each data point $\mathbf{x}^i$. The classifier accomplishes this prediction by computing the decision function for each $\mathbf{x}^i$. This function is computed as follows:

$$\mathbf{w}^\intercal \mathbf{x}^i + b = w_1 x_1^i + \cdots + w_P x_1^P + b \tag{2.4.2}$$

And the predicted binary class $\hat{y}^i$ is learned according to the following rule:

$$\hat{y}^i = \begin{cases} -1, & \mathbf{w}^\intercal \mathbf{x}^i + b < 0 \\ 1, & \mathbf{w}^\intercal \mathbf{x}^i + > 0 \end{cases} \tag{2.4.3}$$

The parameter $\mathbf{w}$ is the learnable weight vector and b is the bias value. The decision boundary of the linear SVM is the set of data points $\mathbf{x}^i$'s for $i \in \{1, 2, \ldots, M\}$ whose decision function $\mathbf{w}^\intercal \mathbf{x}^i + b = 0$. A new data point $\mathbf{x}^*$ can be classified by simply computing $\mathbf{w}^\intercal \mathbf{x}^* + b$. To construct the optimization objective function, which is used to train the linear SVM, we define a loss function (2.4.4).

$$l\left(y^i, \hat{y}^i\right) = \begin{cases} 0, & y^i = \hat{y}^i \\ 1, & y^i \neq \hat{y}^i \end{cases} \tag{2.4.4}$$

Each mislabeled point produces a loss of 1. The training error over M data points is $\sum_{i=1}^{M} l\left(y^i, \hat{y}^i\right)$. The goal is to minimize this loss function while obtaining wide margin discriminators (Figure 3). The slope of

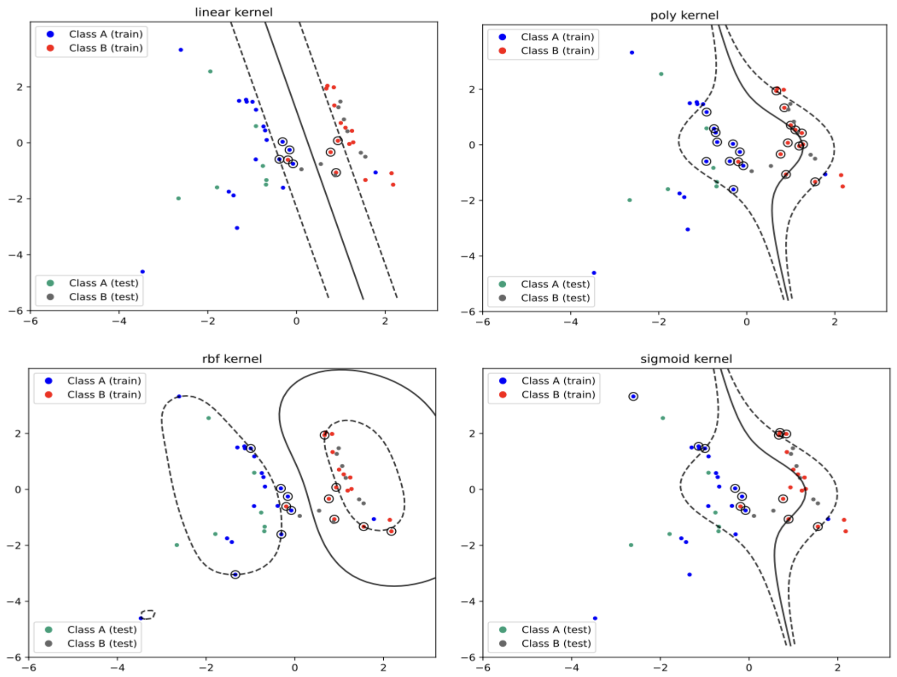


Figure 4: **Illustration of decision boundaries for different kernels.** Four large margins of the SVM classifiers showing the support vectors (data points in circles) that determine the boundaries of these margins (a) for linear kernel, (b) for polynomial kernel (c) radial basis function kernel, and (d) sigmoid kernel. Here the solid line represents the decision boundary and dashed lines represent the margins.

the decision function is equal to the norm $\|\mathbf{w}\| = \sqrt{\mathbf{w}^\intercal \mathbf{w}}$. We want to minimize $\|\mathbf{w}\|$ to get a large margin, i.e., small weight vector $\mathbf{w}$ results in larger margin. The Linear SVM optimization problem is given by (2.4.5).

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{M} l\left(y^i, \hat{y}^i\right) \right\}, \mathit{subject\ to} \min_i |\mathbf{w}^\intercal \mathbf{x}^i| = 1 \tag{2.4.5}$$

We note that in the objective function (2.4.5), we minimize $\frac{1}{2}\|\mathbf{w}\|^2$ and not $\|\mathbf{w}\|$ as the former is differentiable everywhere, while the latter is not differentiable at $\mathbf{w} = 0$. However, the objective function (2.4.5) is difficult to optimize because the loss function is discrete and constructed from zeros and ones. For example, the gradient descent optimization algorithm requires a smooth objective function. To fix this, we revise (2.4.5). We start by defining the so-called Hinge loss (Lee and Lin, 2013) (2.4.6), which is a smooth function.

$$H(y^i, \mathbf{w}^\intercal \mathbf{x}^i + b) = \max\left(0, 1 - y^i \times (\mathbf{w}^\intercal \mathbf{x}^i + b)\right) \tag{2.4.6}$$

When y^i and $\mathbf{w}^\intercal\mathbf{x}^i + b$ have the same sign in (2.4.6), i.e., the linear SVM predicts correctly, then $H(y^i, \mathbf{w}^\intercal\mathbf{x}^i + b) = 0$. When y^i and $\mathbf{w}^\intercal\mathbf{x}^i + b$ have opposite signs, $H(y^i, \mathbf{w}^\intercal\mathbf{x}^i + b)$ increases linearly with $\mathbf{w}^\intercal\mathbf{x}^i + b$. The revised objective function is given by (2.4.7)

$$\min_{\mathbf{w},b} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{M} H(y^i, \mathbf{w}^\intercal\mathbf{x}^i + b) \right\}, \textit{subject to} \min_i |\mathbf{w}^\intercal\mathbf{x}^i| = 1 \tag{2.4.7}$$

If a dataset is not linearly separable, a linear SVM will not generalize well to the training dataset as well as to new data for testing purposes. To fix this problem, we would like to have a nonlinear decision boundary (Figure 4), which extends the linear SVM to a nonlinear SVM. We start by creating a transformation of each $\mathbf{x}^i$ into a higher dimensional space, we will call this transformation $\phi(\mathbf{x}^i)$. For example, if we limit the number of features $P = 2$ in the dataset $\mathbf{x}$, so that each $\mathbf{x}^i$ is $2D$, then a possible transformation of each $\phi(\mathbf{x}^i)$ shown below (2.4.8) in $3D$ is a second-degree polynomial mapping:

$$\phi(\mathbf{x}^i) = \phi\left(\begin{pmatrix} x_1^i \\ x_2^i \end{pmatrix}\right) = \begin{pmatrix} {x_1^i}^2 \\ \sqrt{2}x_1^i x_2^i \\ {x_2^i}^2 \end{pmatrix} \tag{2.4.8}$$

Now, consider two arbitrary $2D$ vectors $\mathbf{a} = (a_1, a_2)$ and $\mathbf{b} = (b_1, b_2)$ in the dataset $\mathbf{x}$, we will apply the second-degree mapping $\phi(\cdot)$ (2.4.8) to the vectors $\mathbf{a}$ and $\mathbf{b}$. Next, we will compute the dot product of the transformed vectors:

$$\begin{aligned} \phi(\mathbf{a})^\intercal\phi(\mathbf{b}) &= \begin{pmatrix} a_1^2 \\ \sqrt{2}a_1a_2 \\ a_2^2 \end{pmatrix}^\intercal \begin{pmatrix} b_1^2 \\ \sqrt{2}b_1b_2 \\ b_2^2 \end{pmatrix} = a_1^2b_1^2 + 2a_1b_1a_2b_2 + a_2^2b_2^2 \\ &= (a_1b_1 + a_2b_2)^2 = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^\intercal \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)^2 = (\mathbf{a}^\intercal\mathbf{b})^2 \end{aligned} \tag{2.4.9}$$

(2.4.9) is called the kernel trick. We will define $(\mathbf{a}^\intercal\mathbf{b})^2$ in (2.4.9) as the function $K(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^\intercal\mathbf{b})^2$ which is also called a second-degree polynomial kernel. In machine learning, a kernel is a function that is capable of computing the dot product $\phi(\mathbf{a})^\intercal\phi(\mathbf{b})$, based only on the original vectors $\mathbf{a}$ and $\mathbf{b}$, without having to compute or even know about the transformation ϕ. The commonly used kernels are listed below:

- Linear: $K(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\intercal\mathbf{b}$
- Polynomial: $K(\mathbf{a}, \mathbf{b}) = (\gamma\mathbf{a}^\intercal\mathbf{b} + r)^N$
- Radial Basis Function: $K(\mathbf{a}, \mathbf{b}) = \exp\left(-\gamma\|\mathbf{a} - \mathbf{b}\|^2\right)$
- Sigmoid: $K(\mathbf{a}, \mathbf{b}) = \tanh\left(\gamma\mathbf{a}^\intercal\mathbf{b} + r\right)$

Mercer's theorem (Murphy, 2012) is often used to determine which function K can be used as a kernel function. According to Mercer's theorem, for a dataset $\mathbf{x} = (\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^M)^\intercal$, if G is a Gram Matrix defined by (2.4.10), and suppose G is positive definite, then each entry $K(\mathbf{x}^j, \mathbf{x}^i)$ in G for $i, j \in \{1, 2, \ldots, M\}$ can be written as $K(\mathbf{x}^j, \mathbf{x}^i) = \phi(\mathbf{x}^j)^\intercal\phi(\mathbf{x}^i)$ for some ϕ that maps $\mathbf{x}$ into another space (possibly a higher dimensional space). We note that we do not have to know ϕ, we only show that the Gram Matrix we obtain from the function K satisfies Mercer's theorem.

$$G = \begin{bmatrix} K(\mathbf{x}^1, \mathbf{x}^1) & \cdots & K(\mathbf{x}^1, \mathbf{x}^M) \\ \vdots & \vdots & \vdots \\ K(\mathbf{x}^M, \mathbf{x}^1) & \cdots & K(\mathbf{x}^M, \mathbf{x}^M) \end{bmatrix} \tag{2.4.10}$$

However, some functions do not satisfy Mercer's theorem but are frequently used as kernel functions such as the Sigmoid kernel because they are known to work well in practice. Using the transformation $\phi(\cdot)$ of each $\mathbf{x}^i$ into a higher dimensional space $\phi(\mathbf{x}^i)$, and setting $w = \sum_{j=1}^{M} \alpha_j \phi(\mathbf{x}^j)$, where α_j's are trainable parameters, we replace the LHS of (2.4.2) an extension of the linear decision function for each $\mathbf{x}^i$ to a nonlinear decision function as follows:

$$\begin{aligned} \mathbf{w}^\intercal \mathbf{x}^i + b &= \left(\sum_{j=1}^{M} \alpha_j \phi(\mathbf{x}^j) \right)^\intercal \phi(\mathbf{x}^i) + b \\ &= \sum_{j=1}^{M} \alpha_j \phi(\mathbf{x}^j)^\intercal \phi(\mathbf{x}^i) + b \\ &= \sum_{j=1}^{M} \alpha_j K(\mathbf{x}^j, \mathbf{x}^i) + b \end{aligned} \tag{2.4.11}$$

2.5 Validation with test set

Validating ML-PETIL is a key factor in assessing its classification performance. In a binary classification task, the confusion matrix (Table 2) records the results of correctly and incorrectly classified examples of each class.

	Positive Prediction	Negative Prediction
Positive Class	True Positive (TP)	False Negative (FN)
Negative Class	False Positive (FP)	True Negative (TN)

Table 2: Confusion matrix for a binary classification task

It is common to use (2.5.1) as a measure of performance of a classification algorithm, however in the case of imbalanced dataset (where we have a majority and a minority class in our dataset) classification accuracy will not distinguish between the number of correctly classified examples of different classes. This may lead to an erroneous conclusion. When we have an imbalanced dataset, the evaluation of a classifiers' performance must be carried out using metrics that considers the distribution of the classes. First, we discuss four metrics we can obtain from Table 2 that measures the classification performance of both positive and negative classes independently:

$$Classification\ accuracy = \frac{TP + TN}{TP + FN + FP + FN} \tag{2.5.1}$$

- True positive rate: $\frac{TP}{TP+FN}$ is the percentage of positive instances correctly classified
- True negative rate: $\frac{TN}{FP+TN}$ is the percentage of negative instances correctly classified

- False positive rate: $\frac{FP}{FP+TN}$ is the percentage of negative instances misclassified
- False negative rate: $\frac{FN}{TP+FN}$ is the percentage of positive instances misclassified

`True positive rate` is usually called the `sensitivity/recall`, `True negative rate` called `specificity`, and `False positive rate` called `1-specificity`. The above metrics are individual metrics, we need to combine metrics of both positive and negative classes. A known way to do this is the evaluation metric called the Receiver Operating Characteristic (ROC) (Nettleman, 1988), this is a trade-off between the benefits `(True positive rate)` and costs `(False positive rate)`. The `Area Under the ROC Curve (AUC)` (Hanley and McNeil, 1982) which provides a measure of a classifier's performance for evaluating which model is better on average. The AUC measure is given by (2.5.2).

$$AUC = \frac{1 + True\ positive\ rate - True\ negative\ rate}{2} \tag{2.5.2}$$

Another significant performance metric is the `F-score` F_β:

$$F_\beta = \frac{(1+\beta^2)(Precision \times True\ positive\ rate)}{\beta^2 \times precision + True\ positive\ rate} \tag{2.5.3}$$

A common choice for β is 1, where equal importance is assigned to the `True positive rate` and the `Precision`, where F_1 is the statistical harmonic mean of the `True positive rate` and the `Precision`.

$$Precision = \frac{TP}{TP + FP} \tag{2.5.4}$$

3 Results

We use demographic, clinical, and biological tumor sample (BTS) features to train ML-PETIL on patients' tumor data and test different predictor dataset to identify various combination of demographic, clinical, and BTS features that yield optimal predictors.

3.1 Data

The retrospective and de-identified patient cohort was selected from H. Lee Moffitt Cancer Center & Research Institute (Aydin et al., 2021). The features in this dataset were categorized as either demographic, clinical, or biological tumor sample (BTS) features.

The demographic characteristics are the following:

1. Surgeon – a categorical data of an unordered list of 7 surgeons.
2. Sex – a categorical data, that categorizes patients as either male or female.
3. Age at Surgery – it takes natural number values.
4. Race – is a categorical data of White, Black, Asian, Native Hawaiian/Pacific Islander, American Indian/Alaskan native, and unknown race.
5. BMI – is a positive real valued data.

6. Ethnicity – a categorical data, that categorizes patients as Hispanic/Latino, Not Hispanic, and Unknown.
7. Surgery – a categorical data, that specifies whether the patient had a Radical Cystectomy or Trans urethral resection of bladder tumor (TURBT).
8. Smoker – this is an ordinal data, it takes values in a finite ordered set: never smoker=0, ever smoker=1, current smoker=2.

The second categorization of features is the clinical features, some of the clinical features are the following:

1. non-muscle invasive bladder cancer (NMIBC) vs muscle-invasive bladder cancer (MIBC) – categorical data, NMIBC=1, MIBC=2.
2. Tumor recurrence within 5 years of neoadjuvant chemotherapy (NAC) – categorical data, No NAC=0, NAC=1.
3. histology including pure urothelial carcinoma, squamous differentiation, plasmacytoid, micropapillary, sarcomatoid, and other variants.

Some biological tumor sample (BTS) features include the following:

1. Number of tumor fragments plated.
2. Tumor digest count.
3. Sample weight of the tumor.
4. CD3 percentage in the tumor digest.
5. CD4 percentage in the tumor digest.
6. CD8 percentage in the tumor digest.
7. MDSC percentage in the tumor digest.
8. CD3 + CD8 percentage in the tumor digest.
9. CD3 + CD8 + PD-1 percentage in the tumor digest.
10. CD3 + CD8 + 41BB percentage in the tumor digest.

The targets in our dataset include:

1. Overall TIL growth
2. TIL growth in the primary tumor
3. TIL growth in the lymph node
4. Total TIL number

5. Overall reactivity

6. The reactivity of TIL from the primary tumor

7. Reactivity of TIL from the lymph node

8. The number of fragments that grew.

In our paper, we will consider overall TIL growth as the target in ML-PETIL. A summary of the demographic and clinical characteristics of the 56 patients' tumor data is presented in Table 3 and Table 4 respectively.

3.2 Constructing a predictor dataset (A demographic, clinical, and BTS predictor)

We utilized a cohort dataset collected at H. Lee Moffitt Cancer Center & Research Institute which contains 56 patient data, and we identified 67 non-overlapping and non-repeating demographic (8), clinical (25), and BTS (34) features. We note that this dataset has some missing datapoints in mostly clinical and BTS features, our goal is to construct a predictor dataset that least sparse in its feature columns and has a robust set of features that produces optimal classification and discrimination of the 'Grew TILs' and 'No TILs' classes. We present simulation results of four predictor dataset.

This predictor is constructed using patients' data consisting of the least sparse features in the three feature categories: demographic, clinical, and BTS features, starting with 25 features – 7 demographic, 15 clinical, and 3 BTS features and after applying the feature selection algorithm (2.2) with repeats and setting a threshold to select the most robust feature (Figure 8), we identified 10 features consisting of 2 demographic, 5 clinical, and 3 biological tumor sample (BTS) features. We trained and tested the classifier using 37 patients' data, (we removed patients missing data). The 10 identified features are the following: `Sample weight of the tumor`, `Number of tumor fragments plated`, `NAC`, `Age at Surgery`, `Tumor digest count`, `pT`, `Surgeon`, `Secondary Histology`, `Histology`. The threshold in Figure 8 selects the optimal number of features with $FIS > 0.04$ in at least 3 of 5 repeats of the feature selection algorithm on the training dataset.

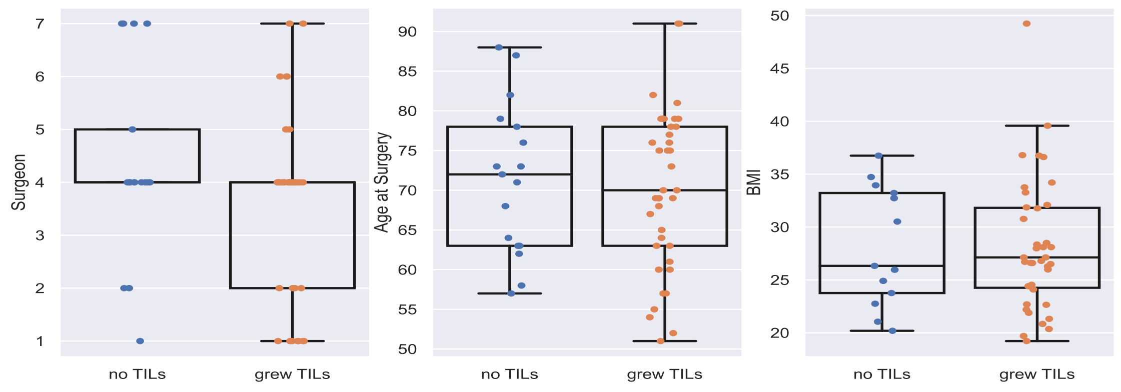


Figure 5: Boxplots of some demographic features showing the distribution of data in the two TIL growth classes

The distribution of the '$GrewTILs'$ and '$NoTILs'$ classes are presented in Figures **??** 4. In Figure 11a, we present the confusion matrix of the optimally selected nonlinear SVM classifier, which we obtain by

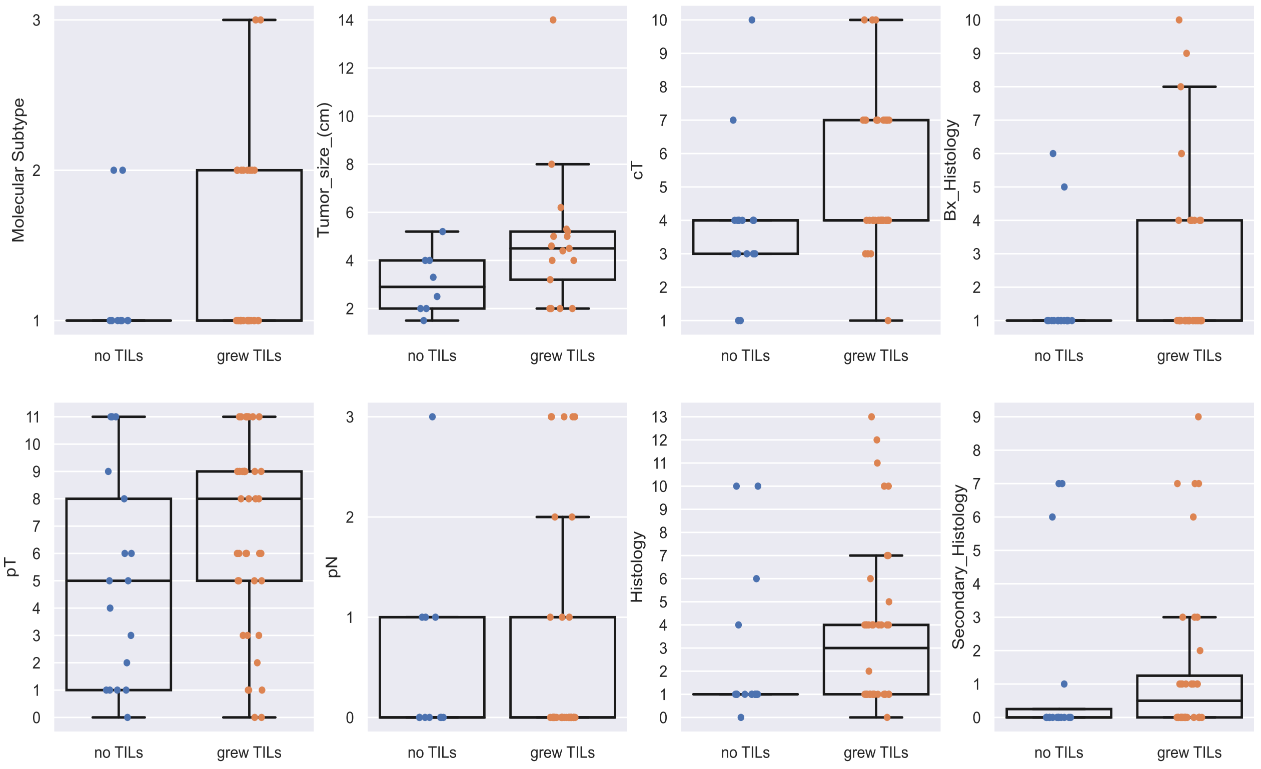


Figure 6: Boxplots of some clinical features showing the distribution of data in the two TIL growth classes

learning the optimal SVM classifier hyperparameter set from a manifold of hyperparameter sets. We use the function '`GridsearchCV`' – a Scikit Learn function to search within the hyperparameter space. We obtain a `classification accuracy` of 0.75, `precision` of 0.778, `sensitivity` of 0.875, `F1 score` of 0.824 and `specificity` of 0.5.

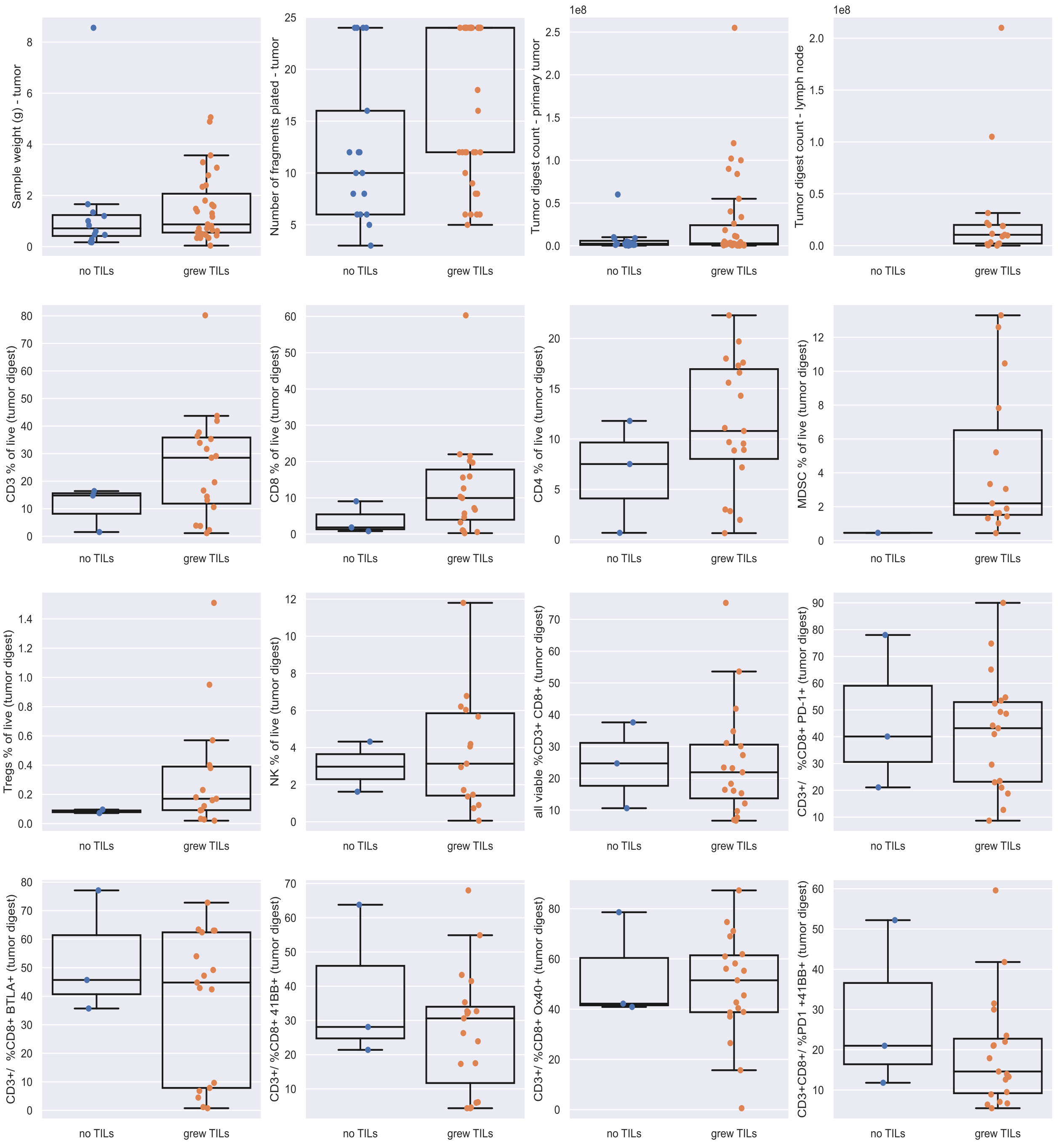


Figure 7: Boxplots of some biological tumor sample (BTS) features showing the distribution of data in the two TIL growth classes

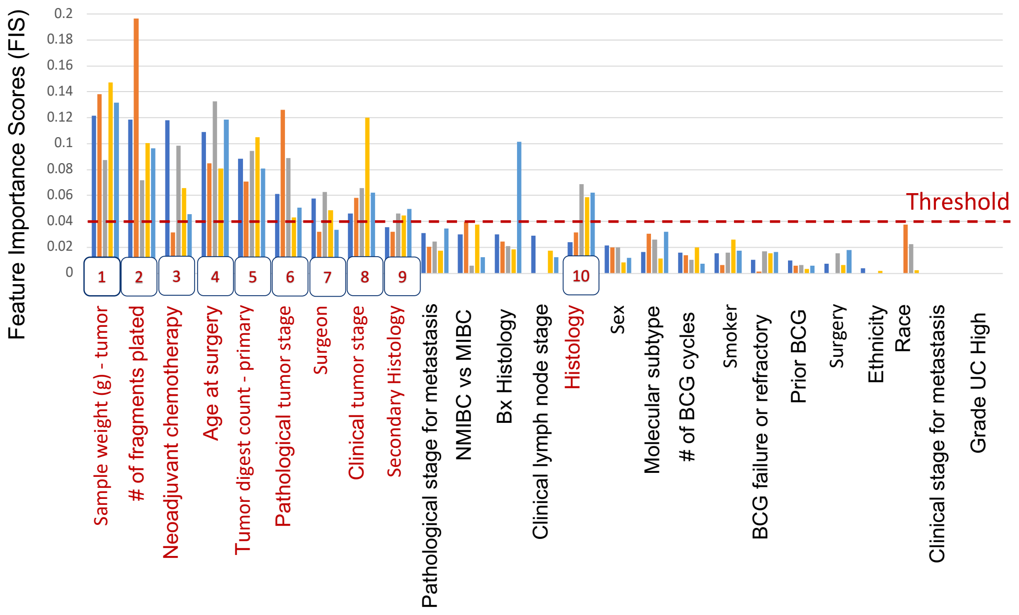


Figure 8: A demographic, clinical and BTS predictor - Random Forest ranking of 25 features using 37 patients' data

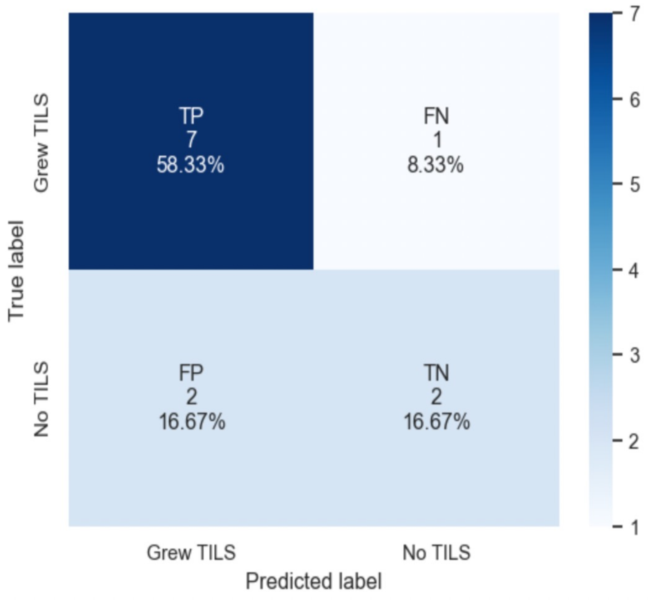


Figure 9: The confusion matrix of the optimally selected nonlinear SVM classifier.

4 Discussion

In this study we developed ML-PETIL – a data-informed machine learning protocol that selects robust features from data and optimally selects classifier by learning the optimal hyperparameter set from an hyperparameter space, where each hyperparameter set correspond to a classifier. This approach produces an optimal discriminator of patients' bladder tumor data into `'Grew TILs'` and `'No TILs'` classes, which can become a useful tool for clinicians when assessing their patient's suitability for the Act-TIL therapy. Analysis of the demographic, clinical, and BTS predictor in section 3.2 demonstrate that the classifier performed well on the `'Grew TILs'` class but performed poorly in the `'No TILs'` class. We see that metrics such as `precision`, `sensitivity`, and `F1 score` have their scores ≈ 0.8 or higher, while the `specificity` metric was 0.5. this mean that our classifier was no better than guessing on the `'No TILs'` class. We note that the Random Forest algorithm does not explore the connectivity between the features when it generates ranks for each feature. We used these ranking to select a low dimensional feature space, which was used to train a nonlinear SVM classifier. We hypothesize that in order to improve the predictability of the predictor presented in section 3.2, we suggest using a feature selection algorithm that considers the connectivity between the features as it builds a nonlinear low dimensional feature space. An example of such algorithm is a manifold learning algorithm-Isomap.

References

Abraham, A. et al. (2014). Machine learning for neuroimaging with scikit-learn. *Front Neuroinform*, 8:14.

Abreu, P. H. et al. (2016). Predicting breast cancer recurrence using machine learning techniques: A systematic review. *Acm Computing Surveys*, 49(3).

Aydin, A. M. et al. (2021). The factors affecting expansion of reactive tumor infiltrating lymphocytes (til) from bladder cancer and potential therapeutic applications. *Front Immunol*, 12:628063.

Besser, M. J. et al. (2010). Clinical responses in a phase ii study using adoptive transfer of short-term cultured tumor infiltration lymphocytes in metastatic melanoma patients. *Clin Cancer Res*, 16(9):2646–2655.

Breiman, L. (1984). *Classification and regression trees*. Belmont, Calif., Wadsworth International Group.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):148–156.

Chapelle, O. et al. (1999). Support vector machines for histogram-based image classification. *IEEE Trans Neural Netw*, 10(5):1055–1064.

Ding, C. and Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *J Bioinform Comput Biol*, 3(2):185–205.

Garapati, S. et al. (2017). Urinary bladder cancer staging in ct urography using machine learning. *Med Phys*, 44(11):5814–5823.

Géron, A. l. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Beijing China ; Sebastopol, CA, O'Reilly Media, Inc.

Guyon, I. et al. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422.

Hanley, J. and McNeil, B. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Lee, C. P. and Lin, C. J. (2013). A study on l2-loss (squared hinge-loss) multiclass svm. *Neural Comput*, 25(5):1302–1323.

Menze, B. et al. (2009). A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics*, 10:213.

	Missing data	Patient Number n(%)	Grew TILs n(%)	No TILs n(%)	P-Value
n		56	39	17	
Surgeon	0				0.274
Surgeon A		8 (14.3)	7 (17.9)	1 (5.9)	
Surgeon B		6 (10.7)	4 (10.3)	2 (11.8)	
Surgeon D		29 (51.8)	20 (51.3)	9 (52.9)	
Surgeon E		4 (7.1)	3 (7.7)	1 (5.9)	
Surgeon F		3 (5.4)	3 (7.7)		
Surgeon G		6 (10.7)	2 (5.1)	4 (23.5)	
Gender	0				1.000
Male		41 (73.2)	28 (71.8)	13 (76.5)	
Female		15 (26.8)	11 (28.2)	4 (23.5)	
Age at Surgery, mean (SD)	0	70.5 (9.8)	70.2 (10.1)	71.4 (9.5)	0.657
Race	0				0.254
White		54 (96.4)	38 (97.4)	16 (94.1)	
Black		1 (1.8)	1 (2.6)		
Native American		1 (1.8)		1 (5.9)	
Ethnicity	0				0.501
Unknown		1 (1.8)	1 (2.6)		
Non-hispanic		53 (94.6)	36 (92.3)	17 (100.0)	
Hispanic		2 (3.6)	2 (5.1)		
BMI, mean (SD)	4	28.2 (6.0)	28.2 (6.2)	28.2 (5.7)	0.990
Surgery	0				0.052
Radical cystectomy		46 (82.1)	35 (89.7)	11 (64.7)	
Transurethral resection		10 (17.9)	4 (10.3)	6 (35.3)	
Smoker	0				0.588
Never smoker		12 (21.4)	9 (23.1)	3 (17.6)	
Ever smoker		34 (60.7)	22 (56.4)	12 (70.6)	
Current smoker		10 (17.9)	8 (20.5)	2 (11.8)	
Patient Number	0				<0.001
Grew TILs		39 (69.6)	39 (100.0)		
No TILs		17 (30.4)		17 (100.0)	

Table 3: Demographic characteristics of 56 bladder cancer patients used to train and validate ML-PETIL

	Missing data	Patient Number n(%)	Grew TILS n(%)	No TILs n(%)	P-Value
n		56	39	17	
Molecular Subtype	19				0.629
Lumil		25 (67.6)	18 (64.3)	7 (77.8)	
Basal		10 (27.0)	8 (28.6)	2 (22.2)	
Double Negative		2 (5.4)	2 (7.1)		
NMIBC_vs_MIBC	0				0.031
NMIBC		12 (21.4)	5 (12.8)	7 (41.2)	
MIBC		44 (78.6)	34 (87.2)	10 (58.8)	
Tumor_size_(cm), mean (SD)	31	4.2 (2.6)	4.8 (2.9)	3.1 (1.3)	0.049
cT	0				0.069
Ta		3 (5.4)	1 (2.6)	2 (11.8)	
T1		8 (14.3)	3 (7.7)	5 (29.4)	
T2		29 (51.8)	21 (53.8)	8 (47.1)	
T3		12 (21.4)	11 (28.2)	1 (5.9)	
T4		4 (7.1)	3 (7.7)	1 (5.9)	
Bx_Histology,	1				0.291
TCC/UC		41 (74.5)	26 (68.4)	15 (88.2)	
Plasmacytoid		1 (1.8)	1 (2.6)		
UC W/ squamous diff		7 (12.7)	7 (18.4)		
Micropapillary		3 (5.5)	2 (5.3)	1 (5.9)	
Small cell		1 (1.8)	1 (2.6)		
UC W/ neuroendocrine		1 (1.8)	1 (2.6)		
UC W/ glandular diff		1 (1.8)		1 (5.9)	
NAC	0				1.000
No NAC		32 (57.1)	22 (56.4)	10 (58.8)	
NAC		24 (42.9)	17 (43.6)	7 (41.2)	
pT	0				0.590
Ta		7 (12.5)	3 (7.7)	4 (23.5)	
Tis		2 (3.6)	1 (2.6)	1 (5.9)	
T0		3 (5.4)	2 (5.1)	1 (5.9)	
T1		4 (7.1)	3 (7.7)	1 (5.9)	
T2		1 (1.8)		1 (5.9)	
T2a		6 (10.7)	4 (10.3)	2 (11.8)	
T2b		8 (14.3)	6 (15.4)	2 (11.8)	
T3a		5 (8.9)	4 (10.3)	1 (5.9)	
T3b		10 (17.9)	9 (23.1)	1 (5.9)	
T4a		10 (17.9)	7 (17.9)	3 (17.6)	
Histology	1				0.586
Benign		2 (3.6)	1 (2.6)	1 (5.9)	
TCC/UC		29 (52.7)	17 (44.7)	12 (70.6)	
Plasmacytoid		4 (7.3)	2 (5.3)	2 (11.8)	
Mixed		1 (1.8)	1 (2.6)		
Clear cell		1 (1.8)	1 (2.6)		
Microcystic		1 (1.8)	1 (2.6)		
SCC		1 (1.8)	1 (2.6)		
UC W/ squamous diff		11 (20.0)	10 (26.3)	1 (5.9)	
UC W/ glandular diff		1 (1.8)	1 (2.6)		
Micropapillary		2 (3.6)	1 (2.6)	1 (5.9)	
Sarcomatoid		2 (3.6)	2 (5.3)		
Prior_BCG	0				1.000
No		44 (78.6)	31 (79.5)	13 (76.5)	
Yes		12 (21.4)	8 (20.5)	4 (23.5)	
Patient Number	0				<0.001
Grew TILs		39 (69.6)	39 (100.0)		
No TILs		17 (30.4)		17 (100.0)	

Table 4: Clinical characteristics of 56 bladder cancer patients used to train and validate ML-PETIL

Mi, H. et al. (2021). Predictive models of response to neoadjuvant chemotherapy in muscle-invasive bladder cancer using nuclear morphology and tissue architecture. *Cell Rep Med*, 2(9):100382.

Morales, A. et al. (1976). Intracavitary bacillus calmette-guerin in the treatment of superficial bladder tumors. *J Urol*, 116(2):180–183.

Murphy, K. P. (2012). *Machine learning : a probabilistic perspective*. Cambridge, MA, MIT Press.

Nembrini, S. (2018). The revival of the gini importance. *Bioinformatics*, 34(21):3711–3718.

Nettleman, M. D. (1988). Receiver operator characteristic (roc) curves. *Infect Control Hosp Epidemiol*, 9(8):374–377.

Peng, H. and Ding, C. (2005). Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans Pattern Anal Mach Intell*, 27(8):1226–1238.

Pilon-Thomas, S. et al. (2012). Efficacy of adoptive cell transfer of tumor-infiltrating lymphocytes after lymphopenia induction for metastatic melanoma. *J Immunother*, 35(8):615–620.

Poch, M. et al. (2018). Expansion of tumor infiltrating lympocytes (til) from bladder cancer. *Oncoimmunology*, 7(9):e1476816.

Powles, T. et al. (2014). Mpdl3280a (anti-pd-l1) treatment leads to clinical activity in metastatic bladder cancer. *Nature*, 515(7528):558–562.

Przedborski, M. et al. (2021). Systems biology informed neural networks (sbinn) predict response and novel combinations for pd-1 checkpoint blockade. *Commun Biol*, 4(1):877.

Quinlan, J. R. (1993). *C4.5 : programs for machine learning*. San Mateo, Calif., Morgan Kaufmann Publishers.

Rosenberg, S. A. and Restifo, N. P. (2015). Adoptive cell transfer as personalized immunotherapy for human cancer. *Science*, 348(6230):62–68.

Shannon, C. E. (1997). The mathematical theory of communication. 1963. *MD Comput*, 14(4):306–317.

Tan, A. C. and Gilbert, D. (2003). Ensemble machine learning on gene expression data for cancer classification. *Appl Bioinformatics*, 2(3):S75–83.

Tokuyama, N. et al. (2022). Prediction of non-muscle invasive bladder cancer recurrence using machine learning of quantitative nuclear features. *Mod Pathol*, 35(4):533–538.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York, Springer.

Washington, P. et al. (2020). Feature selection and dimension reduction of social autism data. *Pac Symp Biocomput*, 25:707–718.

Xu, M. et al. (2017). A deep convolutional neural network for classification of red blood cells in sickle cell anemia. *PLoS Comput Biol*, 13(10):e1005746.

Zhang, X. et al. (2017). Radiomics assessment of bladder cancer grade using texture features from diffusion-weighted imaging. *J Magn Reson Imaging 46(5): 1281-1288*, 46(5):1281–1288.