The Cancer Genome Atlas Kidney Clear Cell Carcinoma

ECE 4783 - Final Project Presentation

Alexander Faché - afache3@gatech.edu

Tuesday April 21, 2020

Content

Clinical Problem Statement

- to develop a decision support model for diagnosis and prognosis of cancer
- follow image processing pipeline for clinical decision making
- application needs
  - early detection
  - accuracy of cancer subtypes
- technical challenges in medical image processing
  - not enough patient data
  - difficult to maintain consistent collection process
    - lighting conditions, stain color
    - high variation across datasets/hospitals

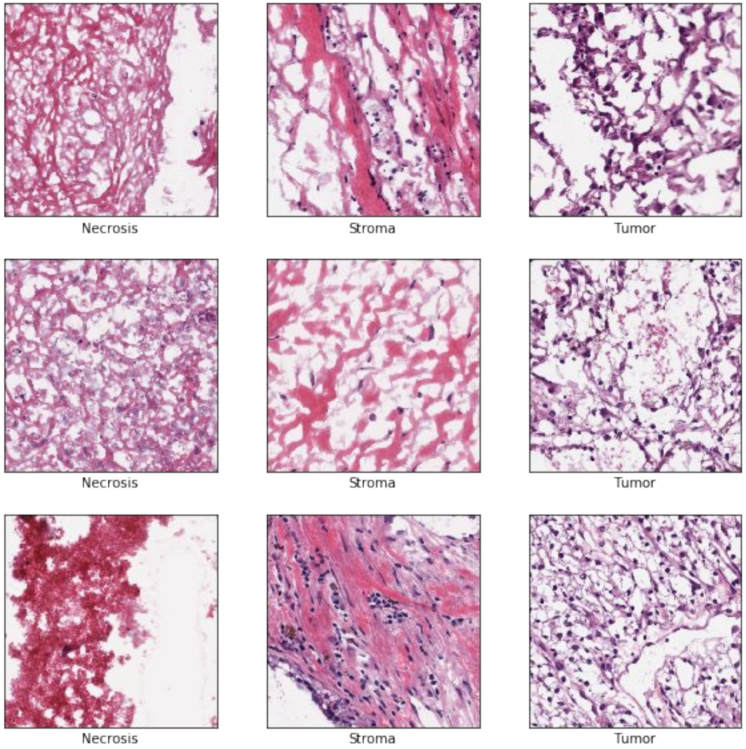
Module 1 - Recap

Initial Dataset

- Necrosis, Stroma, Tumor classes
- 1,000 original whole slide images
    - 20,000 x 40,000 pixels
- "quality control"
    - removal of tissue folding, invalid stains, empty regions of interest
- 512 x 512 pixel sub-section

Undergraduate Dataset

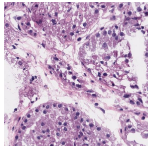
- 100 samples of each class
    - 512 x 512 pixels



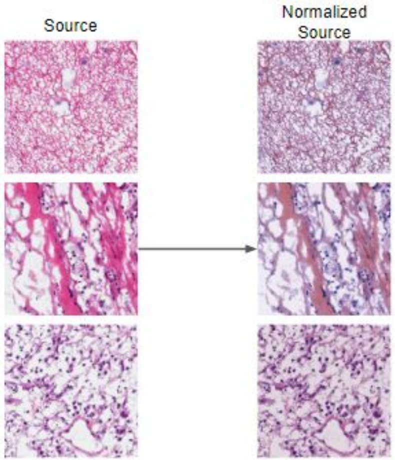
Sampled images that have undergone "quality control".

Flowchart

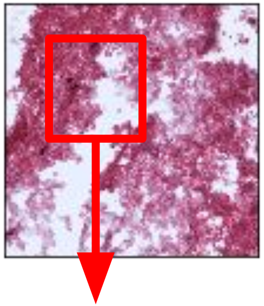
Normalization via Reinhard's Method
- CIELAB color space
- L^* - lightness from black to white
- a^* - lightness from green to red
- b^* - lightness from blue to yellow

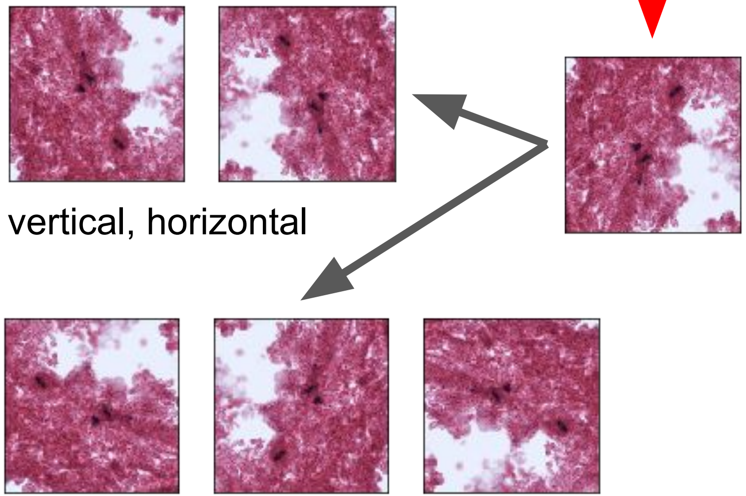


Necrosis target



Data Augmentation
- crop
  - 224 x 224 pixels
- rotation (CCW)
- flip





90°, 180°, 270°

Module 2 - Recap

- feature selection and extraction

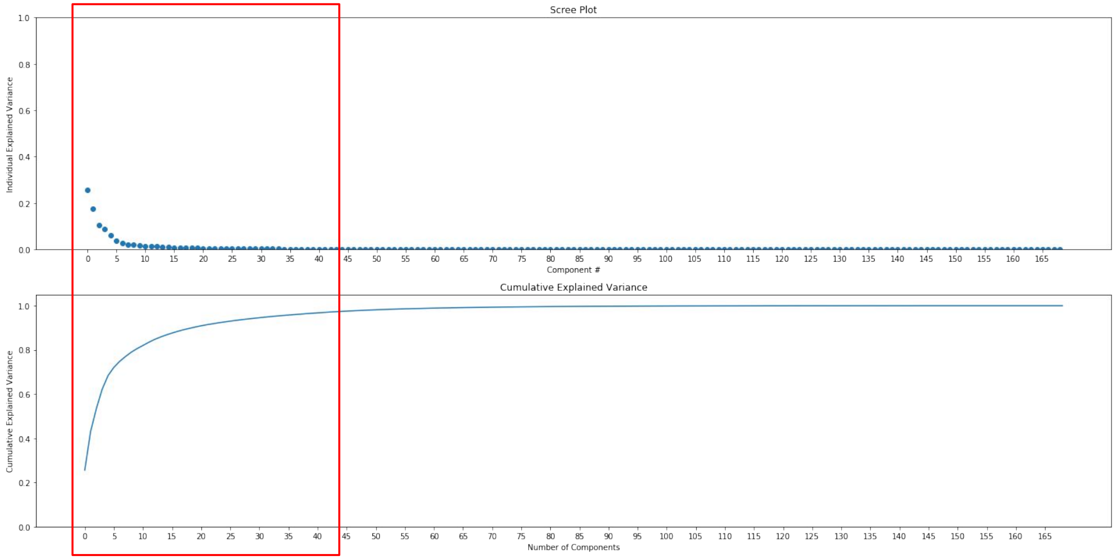
Color Based Features	Texture Based Features	Morphological Based Features
● RGB ● LAB ● HSV ● min, max, mean, variance, skewness, kurtosis for each channel	● GLCM ● contrast, dissimilarity, homogeneity, ASM, energy, correlation, maxprop ● Shannon entropy ● Signal-to-Noise ratio ● histogram ● min, max, mean, variance, skewness, kurtosis for GLCM and histogram properties	● vertical and horizontal symmetry ● center of mass ● percent white and dark pixels ● number of cells ● number of corners ● proportion of edge pixels ● Matlab regionprops ● circle radii ● mean, var, median, mode

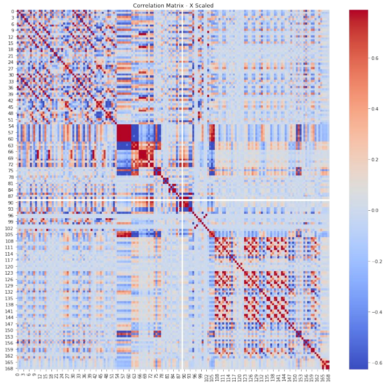
Flowchart*

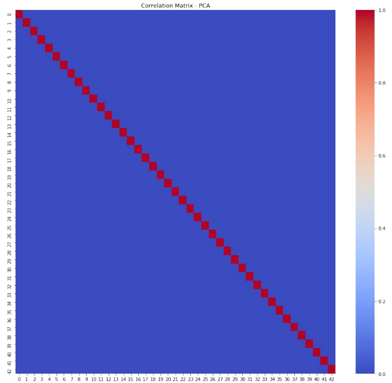
Feature Selection
- literature review
  - color
  - texture
  - morphological

Feature Extraction
- function calls to individually implemented feature extractors

Dimensionality Reduction
- Principal Component Analysis (PCA)
- 97% recovered variance → 43 out of 169 principle components







*Appendix - PCA Visualization

Module 3 - Overview

- literature review of supervised learning methods
- develop classification models
- incorporate cross-validation schemes
- analyze using performance metrics

Literature Review

	Image Classification using Random Forests and Ferns[1]	**Efficient kNN Classification With Different Numbers of Nearest Neighbors[2]**	**Speech Emotion Recognition using Support Vector Machine[3]**	**Multiple Sclerosis Detection Based on Biorthogonal Wavelet Transform, RBF Kernel Principal Component Analysis, and Logistic Regression[4]**
authors	A. Bosch et al.	S. Zhang et al.	M. Jain et al.	S. Wang et al.
model	● random forest, random fern	● kNN, kTree, k*Tree	● support vector machine	● logistic regression
approach	● ROI for object detection ● compare to baseline M-SVM	● optimal k for every test sample using decision tree	● one-against-all (OAA), gender dependent classification	● distinguish between MS or healthy
strengths	● faster to train compared to SVM ● far less computationally expensive	● outperformed competing methods in classification accuracy and running cost	● uses MFCC and LPCC speech feature extractors	● combines biorthogonal wavelet transform, RBF kernel PCA, and logistic regression ● greater sensitivity, accuracy
limitations	● performance is slightly lower than benchmark M-SVM	●	● high variance in speech recording quality	●

Flowchart*

Data
- X (26700 x 43)
- y (26700 x 1)

Split Data into Training and Testing Sets
- 0.70 - 0.30

Training set
- X_train (18690 x 43)
- y_train (18690 x 1)

Select Models
- *KNN*
- *SVM*
- *Random Forest*

Cross-Validation
- test effectiveness of the model
- *K-Fold*
- *Shuffle-Split*

Hyperparameter Optimization
- determine most effective hyperparameters
- *GridSearchCV*

Test
- X_test (8010 x 43)
- y_test (8010 x 1)

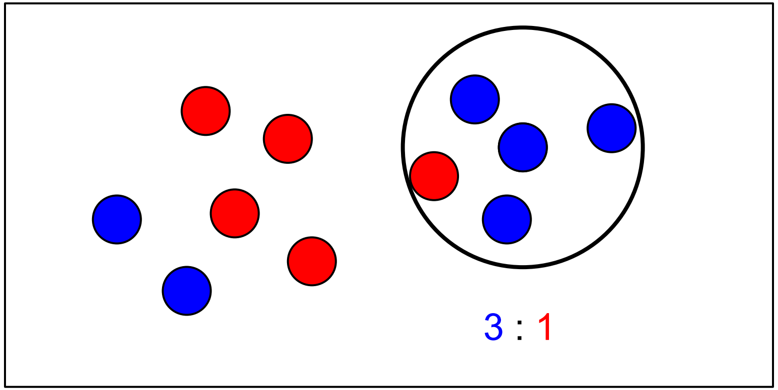
Performance Metrics
- evaluate and verify model performance
- *Confusion Matrix*
- *Accuracy Score*
- *F1-Score*
- *Matthews Correlation Coefficient*
- *Area Under the Curve*

*Appendix - Performance Metrics

K-Nearest Neighbors

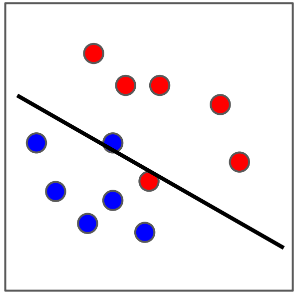
- K = number of neighbors
- distance
  - p = 2 (euclidean distance)
- get distance from every datapoint to current sample
- sort in descending order
- take the top K nearest labels
- class label for new data points is the mode of this subset

Two classes: red, blue
K = 4
New sample: green

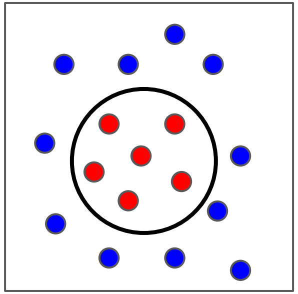


Support Vector Machine

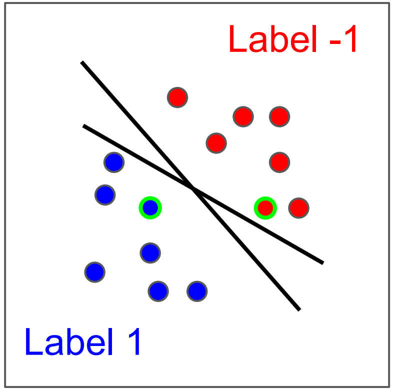
- maximize the margin of the decision boundary
- only works for linearly separable data
  - else use Soft SVM, Kernel
- support vectors define the boundary
- correct classification when
  - $y_i * (x_i * \Theta + b) \geq 0$
- given constrained optimization → lagrangian multiplier
  - $y_i * (x_i * \Theta + b) \geq 1$
    - $|y_i| = 1$
    - $|x_i * \Theta + b| \geq 1$
- optimization makes use of Karush-Kuhn-Tucker (KKT) condition for an inequality constraint
- solve constraint with quadratic programming → margin



Soft SVM



Kernel



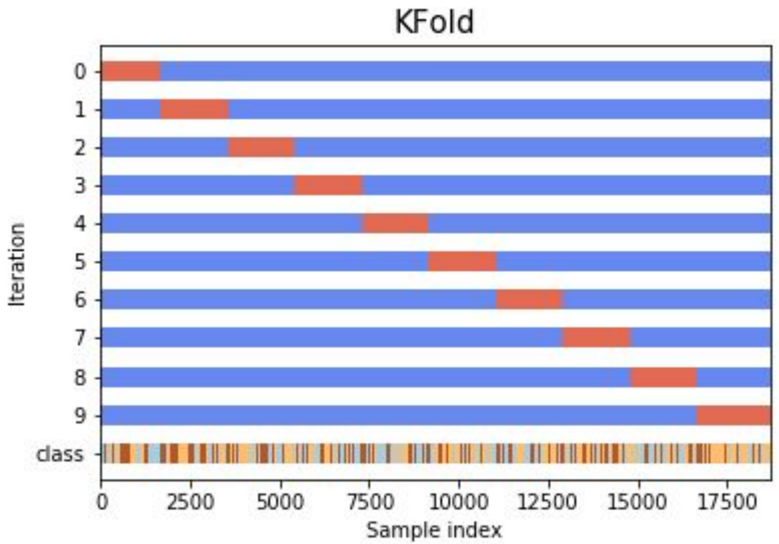
Random Forest = Bagged Random Decision Trees*

- Entropy
  - M classes
  - $H = -\Sigma\{P_i * \log_2 P_i\}|_{i=1:M}$
- Information Gain
  - $IG = H - (H_L * P_L + H_R * P_R)$
- Decision Tree
  - split dataset recursively based on feature and feature value with highest information gain
  - create subsets until leaf nodes represent prediction of class label
- Bagging - **B**ootstrap **Agg**regat**ing**
  - train classifier on subset of training set
  - final classifier uses all sub classifiers to make final prediction
    - B classifiers
    - $f(x) = \text{majority}\{f_j(x)\}|_{j=1:B}$

*Appendix - Random Forest Example

Cross-Validation Schemes - K-Fold*

- test the effectiveness of a ML model before hyperparameter optimization and training
  - split dataset into K subsets
  - each subset is iteratively used as test set
  - estimator is trained on other K-1 subsections
  - estimator is tested on kth subset
  - accuracy score is recorded



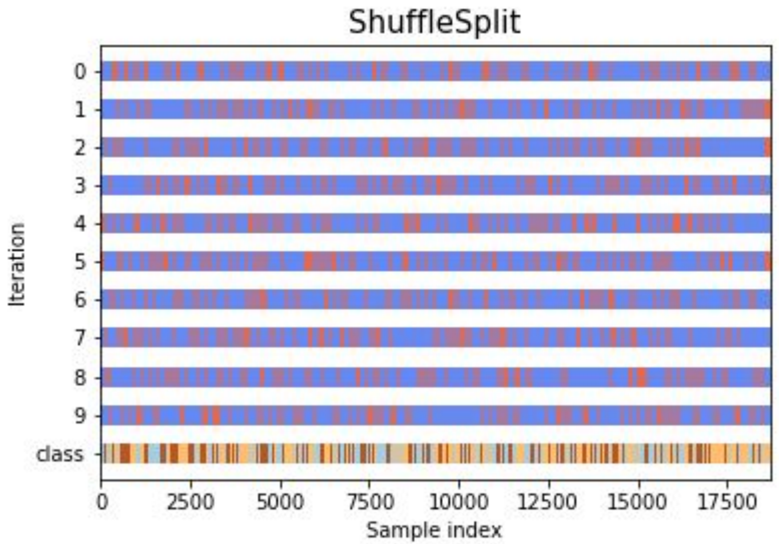
Sample indices. Train, Test.

```python
from sklearn.model_selection import KFold

def cross_validation_KFold(clf, X_train, y_train, title, number_splits=10, plot=True):
    kf = KFold(n_splits=number_splits, shuffle=False)

    accuracy = []
    for train_index, test_index in kf.split(X_train):
        clf = clf.fit(X_train[train_index, :], y_train[train_index])
        y_train_prediction = clf.predict(X_train[train_index, :])
```


*Appendix - K-Fold

Cross-Validation Schemes - Shuffle Split*

- select random indices of training set



Sample indices. Train, Test.

```python
from sklearn.model_selection import ShuffleSplit

def cross_validation_ShuffleSplit(clf, X_train, y_train, title, number_splits=10, plot=True):
    ss = ShuffleSplit(n_splits=number_splits, test_size=0.25)

    accuracy = []
    for train_index, test_index in ss.split(X_train):
        clf = clf.fit(X_train[train_index, :], y_train[train_index])
        y_train_prediction = clf.predict(X_train[train_index, :])
```


*Appendix - Shuffle Split

Hyperparameter Tuning

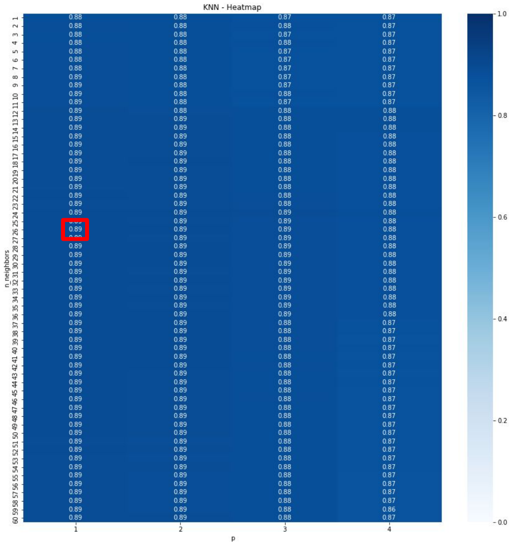
- Exhaustive Grid Search
  - generates estimators from given hyperparameter list
  - returns estimator with highest score

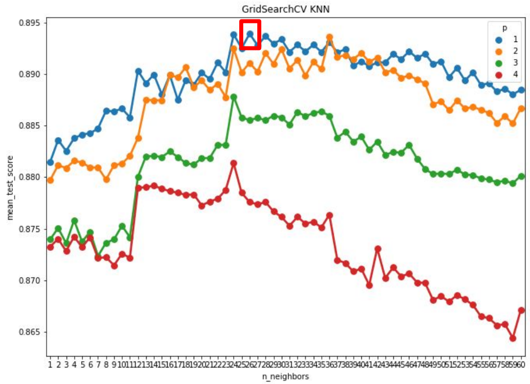
```
from sklearn.model_selection import GridSearchCV

hyperparameters = {...}

clf = GridSearchCV(estimator=..., param_grid=hyperparameters, cv=10, verbose=2, refit=True)
clf = clf.fit(X_train, y_train)
best_estimator = clf.best_estimator_
```

KNN	SVM	Random Forest
n_neighbors: number of neighbors to use	*C*: regularization parameter	*max_depth*: maximum depth of the tree
p: power parameter for the Minkowski metric	γ: kernel coefficient for radial basis function	*min_samples_leaf*: minimum number of samples required to split an internal node

Results - K-Nearest Neighbor

```
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier(n_neighbors=26, p=1)
clf = clf.fit(X_train, y_train)

y_train_prediction = clf.predict(X_train)
y_test_prediction = clf.predict(X_test)
```





	0	1	2
0	6193	10	67
1	47	6100	63
2	22	3	6185

Training Set

```
Accuracy Score: 0.9887
F1-Score: 0.9887
Matthews Correlation Coefficient (MCC): 0.9830
Area Under the Curve (AUC): 0.9830
```

	0	1	2
0	2603	22	45
1	185	2249	146
2	197	25	2538

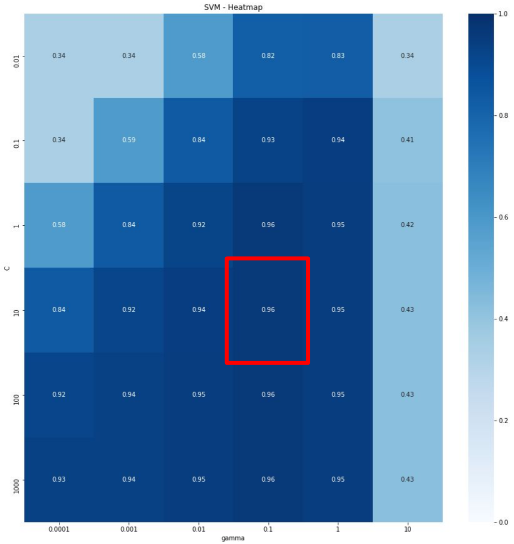
Testing Set

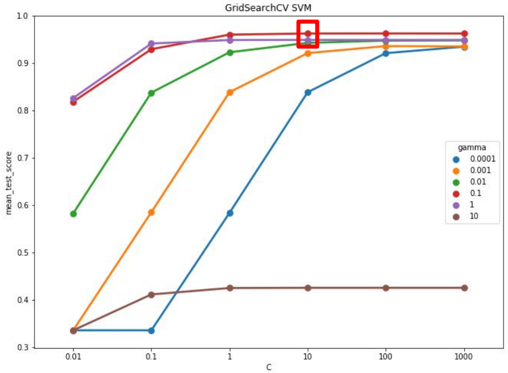
```
Accuracy Score: 0.9226
F1-Score: 0.9226
Matthews Correlation Coefficient (MCC): 0.8857
Area Under the Curve (AUC): 0.8857
```


*Appendix - Decision Boundary Visualization

Results - Support Vector Machine

```python
from sklearn.svm import SVC

clf = SVC(kernel='rbf', C=10, gamma=0.1)
clf = clf.fit(X_train, y_train)

y_train_prediction = clf.predict(X_train)
y_test_prediction = clf.predict(X_test)
```





Training Set

	0	1	2
0	6270	0	0
1	0	6210	0
2	6	0	6204

Accuracy Score: 0.9997
F1-Score: 0.9997
Matthews Correlation Coefficient (MCC): 0.9995
Area Under the Curve (AUC): 0.9995

Testing Set

	0	1	2
0	2603	33	34
1	36	2526	18
2	122	36	2602

Accuracy Score: 0.9652
F1-Score: 0.9653
Matthews Correlation Coefficient (MCC): 0.9480
Area Under the Curve (AUC): 0.9480

Results - Random Forest*

```python
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, max_depth=18, min_samples_leaf=14)
clf = clf.fit(X_train, y_train)

y_train_prediction = clf.predict(X_train)
y_test_prediction = clf.predict(X_test)
```

Training Set

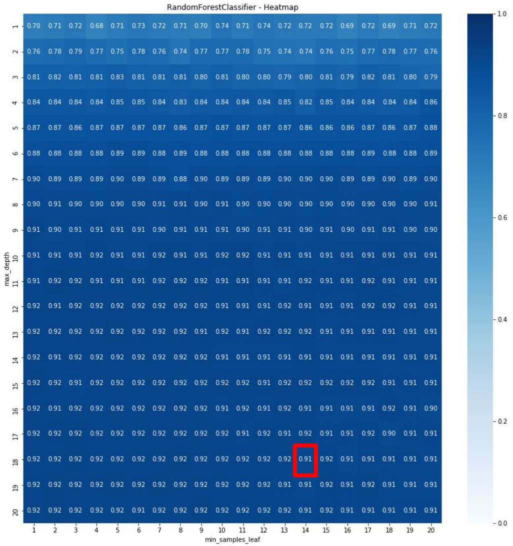
	0	1	2
0	6270	0	0
1	0	6203	7
2	0	0	6210

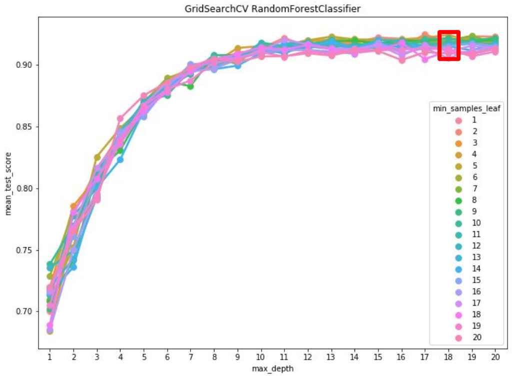
```
Accuracy Score: 0.9996
F1-Score: 0.9996
Matthews Correlation Coefficient (MCC): 0.9994
Area Under the Curve (AUC): 0.9994
```

Testing Set

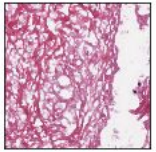
	0	1	2
0	2519	57	94
1	93	2446	41
2	198	98	2464

```
Accuracy Score: 0.9275
F1-Score: 0.9277
Matthews Correlation Coefficient (MCC): 0.8917
Area Under the Curve (AUC): 0.8917
```

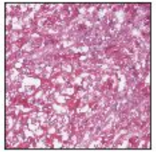




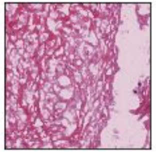
Demo



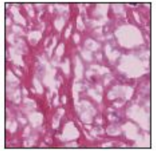
Test Image

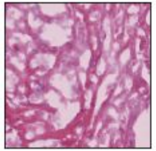


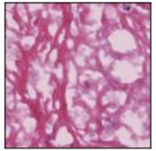
Target Image

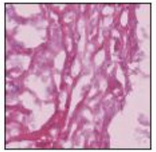


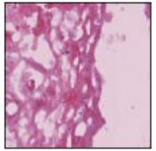
Normalized Image



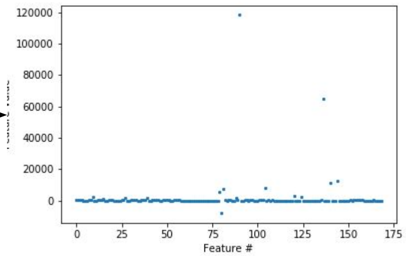


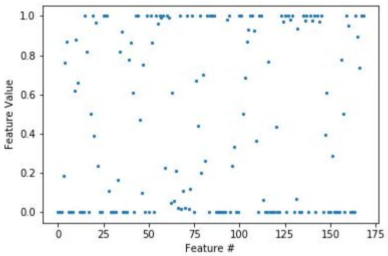






Subsections





169 original features of top subsection	169 scaled features of top subsection.

Pass through PCA to reduce to 43 principle components.

Input features into KNN, SVM, and Random Forest classifiers.

	Subsection 1	Subsection 2	Subsection 3	Subsection 4	Subsection 5
KNN					
SVM					
Random Forest					

Conclusion and Future Work

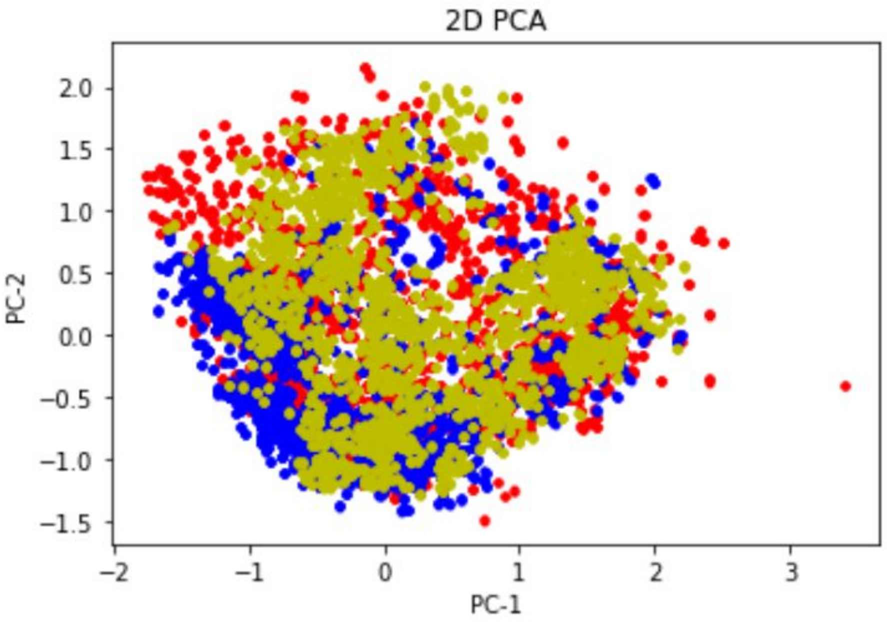
- Hyperparameter Tuning
  - KNN was the simplest with only 1 true parameter
- Training time
  - KNN and Random Forest were more efficient
- Accuracy
  - SVM was slightly more accurate than KNN and Random Forest
- Ensemble learning
  - combine KNN, SVM, Random Forest, etc
- CNN
  - feature extraction
  - prediction

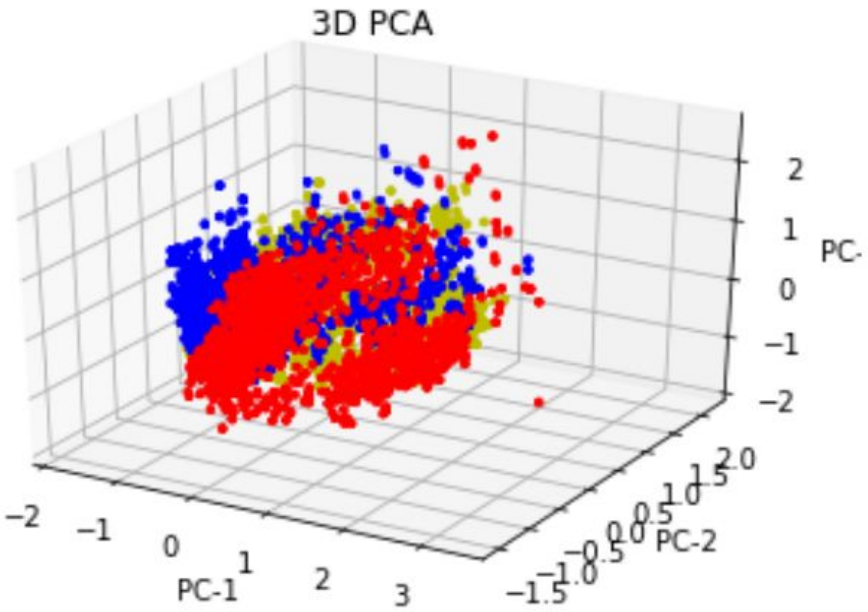
Metric (test set)	KNN	SVM	Random Forest
Accuracy	0.9226	0.9652	0.9220
F1-Score	0.9226	0.9653	0.9222
MCC	0.8857	0.9480	0.8838
AUC	0.8857	0.9480	0.8838

References

1. A. Bosch, A. Zisserman and X. Munoz, "Image Classification using Random Forests and Ferns," 2007 IEEE 11th International Conference on Computer Vision, Rio de Janeiro, 2007, pp. 1-8.
2. S. Zhang, X. Li, M. Zong, X. Zhu and R. Wang, "Efficient kNN Classification With Different Numbers of Nearest Neighbors," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 5, pp. 1774-1785, May 2018.
3. Manas Jain, Shruthi Narayan, Pratibha Balaji, Bharath K P, Abhijit Bhowmick, Karthik R, "Speech Emotion Recognition using Support Vector Machine," 2020, [http://arxiv.org/abs/2002.07590 arXiv:2002.07590].
4. S. Wang et al., "Multiple Sclerosis Detection Based on Biorthogonal Wavelet Transform, RBF Kernel Principal Component Analysis, and Logistic Regression," in IEEE Access, vol. 4, pp. 7567-7576, 2016.

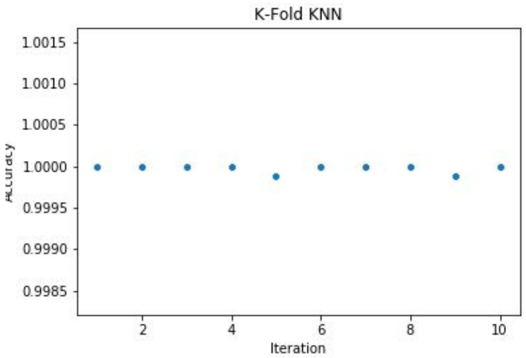
Appendix - PCA Visualization

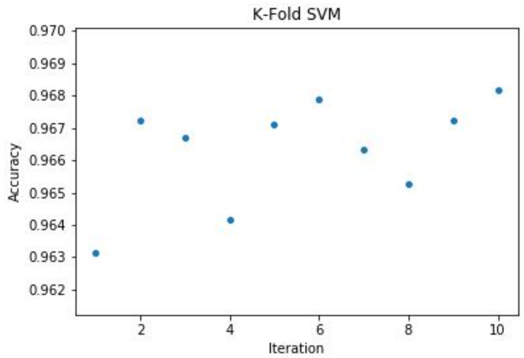


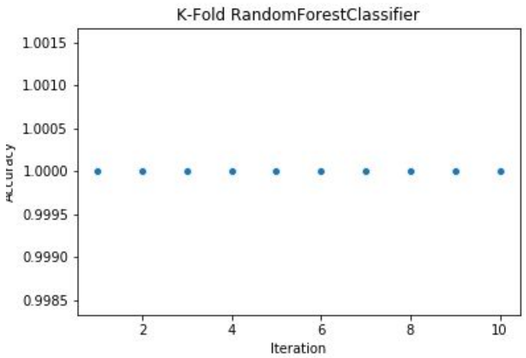


Necrosis, Stroma, Tumor

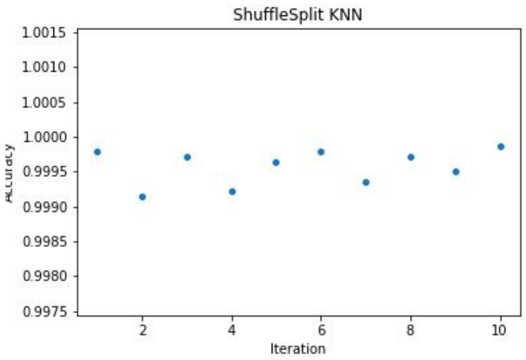
Appendix - K-Fold

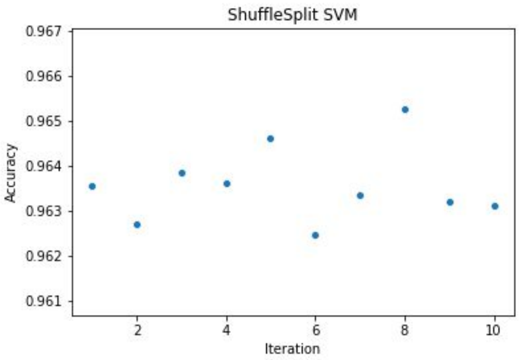


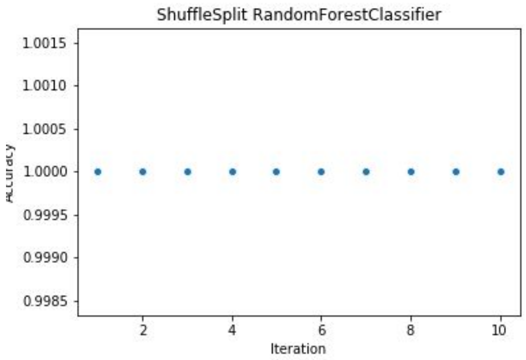




Appendix - Shuffle Split



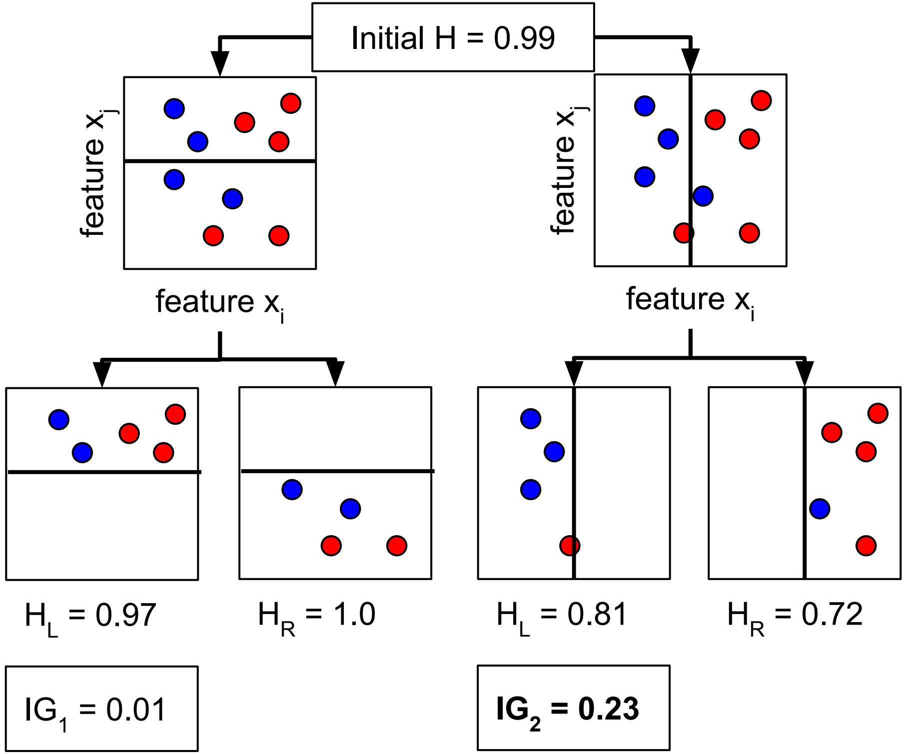




Appendix - Performance Metrics

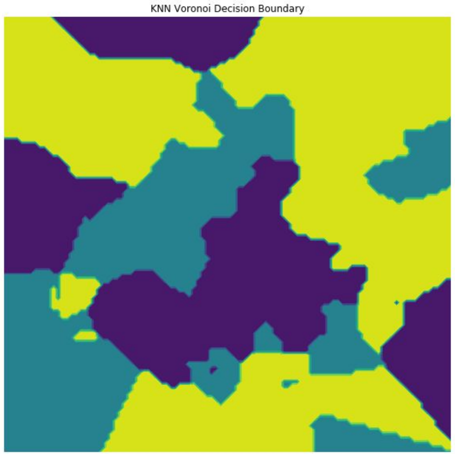
Confusion Matrix	**Accuracy Score**	**F1-Score**	**Matthews Correlation Coefficient**	**Area Under the Curve**
Summary of prediction results among each class for a classification problem.	Fraction of correctly classified labels.	F1 = 2*(precision*recall) / (precision + recall)	Balanced measure taking into account TP, TN, FP, FN.	Trapezoidal rule. TPR vs FPR
Return Values				
Confusion matrix $C_{ij.}$ Observations in group i and predicted in group j.	[0, 1] 0: worst accuracy 1: best accuracy	[0, 1] 0: worst precision and worst recall 1: best precision and best recall	[-1, +1] -1: inverse prediction 0: random prediction +1: perfect prediction	[0, 1] 0: worst performance 1: perfect performance

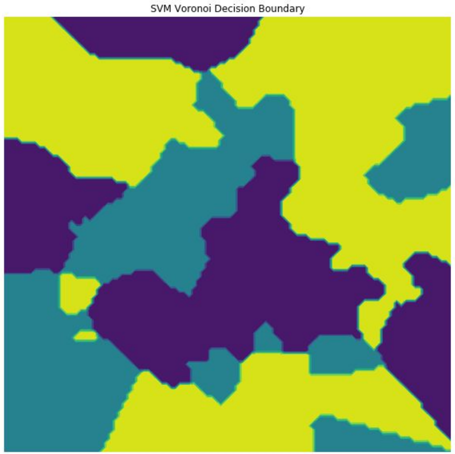
Appendix - Random Forest Example



Select right path ($IG_2 > IG_1$) with feature x_i and the value of the vertical line to use for current split.

Appendix - Decision Boundary Visualization





t-SNE is used to visualize high dimensional data and decision boundaries. Voronoi decision boundaries for KNN, SVM, and Random Forest are shown. Due to computation constraints, a subset of the training set was used for this visualization.