Comparing classification models in order to determine prognosis from the MSI subtype

Group 88: Bouzetos Emmanouil, Dwamena Sedinam Barbara,
Mazumder Sreejita, Walfenzao Alexarae

March 31, 2023

Abstract

Colorectal cancer (CRC) is the third most common form of cancer and there is a need for early identification techniques to begin treatment. Developing a genome profile for CRC would benefit from incorporating biological and machine learning techniques to predict a class for a new dataset. Analysis of machine learning algorithms accuracy in predicting severity of cancer in a new genome is reviewed in this study. Important features were selected by recursive feature elimination, and training was conducted with top 200 features. Keras neural network used to identify hyperparameters. Machine learning models Random Forest, Naive Bayes', Logistic Regression, and Support Vector Machine output were compared for accuracy.

1 Introduction

Colorectal cancer (CRC) is globally the third most common, and second most deadly, form of cancer (Lakbir et al., 2022). CRC has been broadly classified into 4 molecular subtypes; CMS 1(augmented for inflammatory/immune genes), CMS 2(canonical), CMS 3(metabolic) and CMS 4(mesenchymal) (Fontana et al.,2019). Some CRC cases never develop polyposis which can lead to a late diagnosis. When physical phenotypes are inconsistent as a predictor, genome sequence analysis is often used to identify patterns in cancer subtypes. The prediction of cancer subtype requires developing a profile with associated genes of interest and features. Identifying which specific cancer subtypes have features which can be useful in predicting the prognosis of a cancer patient is a challenge.

For instance, hereditary CRC is non-polyposis forming but is also characterized by microsatellite instability (MSI) regions in tumors (Dekker et al., 2019). MSI are caused by DNA mismatch repair (MMR) system impairment and found in 15-20% of all CRC cases (Dekker et al., 2019, Lakbir et al., 2022). Mutations in MMR pathways and polymerases such as DNA polymerase- (POLE) or DNA polymerase- catalytic subunit 1 (POLD1) can lead MSI regions of hypermutated short repeat regions of DNA (Lakbir et al., 2022; Kawakami et al., 2015, Li et al., 2021). MSI tumors can be classified into two groups, high microsatellite instability (MSI-H) and low microsatellite instability (MSI-L) (Kawakami et al., 2015).

A quantifiable feature of MSI tumors is the number of single nucleotide variants, this measure is the tumor mutation burden (TMB) (Lakbir et al., 2022). MSI status is determined by the number of microsatellite markers that show instability, in MSI-H two or more markers show instability and MSI-L showing one marker. However, 80% of CRC tumors present with chromosomal instability instead of deficient MMR systems (Lakbir et al., 2022). These tumors contain proficient MMR systems known as microsatellite stable (MSS) and respond differently to chemotherapy treatment (Kawakami et al., 2015). Of note, patients with MSS and MSI-L tumors have a worse prognosis than patients with tumors containing MSI or deficient MMR systems (Lakbir et al., 2022; Kawakami et al., 2015).

CRC diagnosis has increased over the past years, in part thanks to increased screening and sequence analysis. Optimizing this process with machine learning methods will help increase the ability to diagnose patients faster and begin treatment earlier. It is a challenge to produce a good prediction model when the dataset has many features as mentioned in the previous paragraphs. A good feature selection method can help avoid overfitting and in general improve classification performance by discarding inputs that do not contribute to prediction accuracy of the model. Prediction accuracy is dependent on how well a model can discern which genes are of interest for diagnosis. Therefore to

increase prediction accuracy correlation methods are often used to eliminate genes which cause noise and do not help. Those correlated features become a class predictor with a classifier weight. Ranking is then based on weighted voting, which helps eliminate genes ineffective for discrimination (Guyon et al., 2002). When the classifier performs well those inputs with the largest weights correspond to the most informative features.

To further increase feature selection of high dimensional data, it can be run through a Random Forest classifier. This method will generate decision trees but each split is generated by independently identically distributed random vectors (Breiman 2001). This removes any correlations between features and improves prediction accuracy. The main objective of this paper is to analyze how to classify the class labels worse prognosis and good prognosis for colorectal cancer with machine learning models. This will be conducted by comparing Naive Bayes' Classifier , Logistic Regression and Support Vector Machine (SVM) accuracy, precision, and recall. We will also use Keras deep learning neural network in TensorFlow platform to improve our models.

2 Methods

2.1 Dataset

In this analysis, previously collected genome data from the Cancer Genome Atlas (TCGA) colorectal cancer dataset was used. The dataset consisted of 33379 gene counts from 462 identified colorectal cancer samples. Next to this large gene count dataset, quantitative metrics were provided for 461 genes of interest. These metrics were microsatellite instability (MSI status), tumor break load (TBL), tumor mutational burden (TMB), fraction genome altered (Nguyen et al., 2022), aneuploidy score. Additionally to these metrics the mutational status of frequently mutated genes TP53, KRAS, BRAF, APC, TTN were also labeled. The data was split by random state before analysis into three groups; all groups contained 33376 gene counts, from varying sample sizes the training set contained 348, validation set 44, and test set 43 samples. Further classification was conducted based on MSI status, the accumulation of insertions or deletions errors of microsatellite repeat sequences in cancer cells. MSS and MSI-L were grouped together as both cause worse prognosis, from here on labeled MSS. Additionally, MSI-H labeled solely as MSI.

2.2 Preprocessing of Count Data

During preprocessing any samples containing zero counts were removed from the dataset. Samples with mutations in POLE or POLD1 have high TMB and low TBL, however MSS samples often have variation in TBL (Lakbir et al., 2022). Therefore, POLE3 and POLD2 were removed from the dataset as possible outliers. Next, the counts were normalized using a technique called library size normalization. The last step was to remove the samples from a large count dataset with zero MSI values. The final dataset contained 435 genes and 33377 samples. All preprocessing was conducted using python version xxx.

2.3 Feature Selection

The TCGA data contains multiple dimensions in feature space and requires a feature selection method to assist classification accuracy (Guyon et al., 2002). Recursive feature elimination (RFE) with a decision tree classifier was used for feature selection with a feature range of 500 features. This was built with DecisionTreeClassifier function from sklearn.tree and RFE function from sklearn.featureSelection Python packages. Features are ranked by correlating each feature to a preprocessed subset of features at a threshold of 0.3. The correlation coefficient for each feature becomes a class predictor with a classifier weight. The largest weights correspond to the most informative features. The features with the lowest weights are removed for each iteration and the decision tree classifier was retrained using the reduced feature set. This process was repeated until the desired number of features reached 500. To evaluate the performance of the selected features, the validation set was used to test the decision tree classifier trained on the reduced feature set.

Hidden Layers (no of neurons layer1,no of neurons layer2...)	10,	5,	10,5	20,	20,5	20,10
Activation method	relu	tanh	sigmoid			
Optimizer	sgd	adam	RMSprop			
Learning Rate	0.0001	0.001	0.01			

Table 1: Hyperparameters for cross validation

2.4 Models

Four models were used to analyze performance classifying the TCGA dataset accurately; Naive Bayes' Classifier , Logistic Regression,SVM Neural Network. First, an additional classification step where the top 200 features were classified by Random Forest model. This model generated a random ensemble of trees, each tree's growth was guided by a random vector which is independent of all other random vectors and dispersed uniformly (Breiman, 2001). Therefore the most popular class at input x is voted on by the trees. The output was the top 200 features which were selected for further modeling with Keras. Keras' artificial neural network employs TensorFlow open source machine learning library to explore and develop machine learning models (Abadi et al., 2016).

These 200 features were run in the Naive Bayes' probabilistic classifier method, modeling the distribution of features given a class. Next step was to run the top features through SVM, to classify the two classes in n-dimensional space. Then, Logistic Regression for binary classification based on input characteristics,given the class labels. Each model was trained and validated before running on the test set. The models used are open access Python packages; randomForest function from the sklearn.ensemble module; the keras function in the TensorFlow package; the GaussianNB function from the sklearn.naive_bayes module, the LogisticRegression function from the sklearn.linear_model module; the SVM function from the sklearn module. The algorithms are attached in the supplementary material.

2.4.1 Hyperparameter Selection for Neural Network

In this project, resampling was used to perform hyperparameter tuning for a neural network model using cross-validation. Specifically, parameter sampling was used to generate a grid of hyperparameter combinations to search over, and then evaluate each combination using a cross-validation procedure with 5 folds.

The number of hyperparameter combinations to sample is determined by multiplying the number of options for each hyperparameter together. In this case, the number of combinations is the product of the number of options for 'hidden_layers', 'activation', 'optimizer', and 'learning_rate', which is 6 * 3 * 3 * 3 = 162. The hyperparameters what were used are described in the table 1

For each hyperparameter combination, the 'build_model' function is called to construct a neural network model with the specified hyperparameters. The 'evaluate_model' function is then called to evaluate the model using cross-validation. Within 'evaluate_model', the cross-validation is performed by splitting the training data into 5 folds using stratified k-fold, and training the model on 4 folds while evaluating on the remaining fold. The process is repeated for each fold, and the average accuracy score across all folds is returned. The code keeps track of the hyperparameter combination that achieves the best cross-validation score, and reports that combination as the optimal set of hyperparameters.

For each run of the code, a different set of hyperparameters will be sampled from the hyperparameter grid, and the cross-validation will be performed with different folds. This can lead to different optimal hyperparameters being selected, and different accuracy scores being obtained. However, the overall process of hyperparameter tuning using cross-validation remains the same. The scores for each run with their comparing hyperparameter were used to generate bar charts with error bars to compare the mean and standard deviation of validation accuracy scores for different combinations of hyperparameters for further analysis and can be seen in the supplementary section. The hyperparameters that achieved the highest score are passed to the build model function to create the final model.

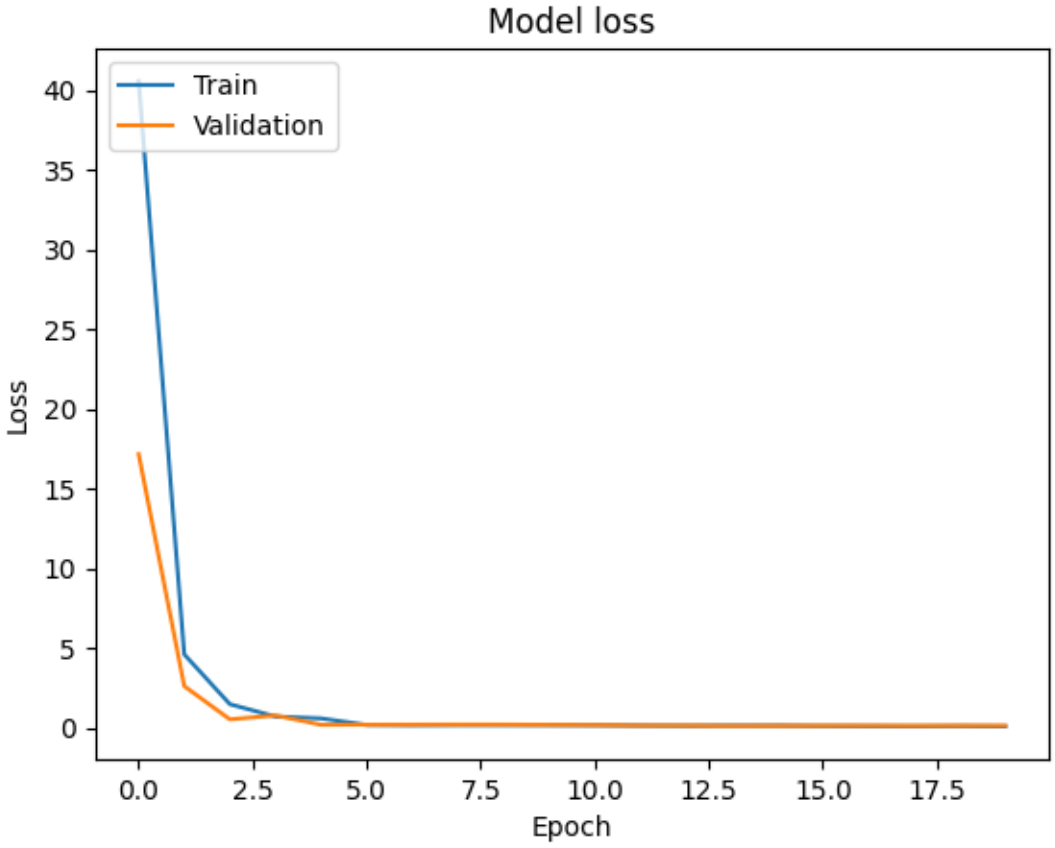


Figure 1: Neural Network Loss During Training

2.4.2 Neural Network

The top 200 features were trained in a deep learning neural network to select the best hyperparameters. The final hyperparameters that were selected from the process described in 2.4.1 were: optimizer: adam, learning rate: 0.01, hidden layers: (20,), activation: ReLu. After the construction of the model it was trained with the full set of X_train and y_train datasets. A selection of 20 epochs were made as it offered a good balance between training and overfitting.The validation datasets were used during the fitting process to provide metrics of validation accuracy and validation loss to explore if the model was being trained correctly and thus avoiding issues such as underfitting and overfitting. The curves of loss and validation loss as shown in the figure 1 indicate that the model was trained correctly with some minimal overfitting.

2.5 Evaluation Metrics

In this project, resampling was used to perform hyperparameter tuning for a neural network model using cross-validation. Specifically, the code uses parameter sampling to generate a grid of hyperparameter combinations to search over, and then evaluates each combination using a cross-validation procedure with 5 folds.

The number of hyperparameter combinations to sample is determined by multiplying the number of options for each hyperparameter together. In this case, the number of combinations is the product of the number of options for 'hidden_layers', 'dropout_rate', 'activation', 'optimizer', and 'learning_rate', which is 6 * 1 * 3 * 3 * 3 = 162.

For each hyperparameter combination, the 'build_model' function is called to construct a neural network model with the specified hyperparameters. The 'evaluate_model' function is then called to evaluate the model using cross-validation. Within 'evaluate_model', the cross-validation is performed by splitting the training data into 5 folds using stratified k-fold, and training the model on 4 folds

		positive	negative
Predicted	positive	True Positive	False Positive
	negative	False Negative	True Negative

Table 2: Classification evaluation confusion matrix

while evaluating on the remaining fold. The process is repeated for each fold, and the average accuracy score across all folds is returned.

The code keeps track of the hyperparameter combination that achieves the best cross-validation score, and reports that combination as the optimal set of hyperparameters.

For each run of the code, a different set of hyperparameters will be sampled from the hyperparameter grid, and the cross-validation will be performed with different folds. This can lead to different optimal hyperparameters being selected, and different accuracy scores being obtained. However, the overall process of hyperparameter tuning using cross-validation remains the same.

The performance of each model was compared by calculated accuracy, precision, recall, and F1 score. Figure 3a displays the classification evaluation metric to determine a score for accuracy, precision, and recall. Together these metrics can help evaluate the output of a model. With balanced data accuracy score reports the proportion of true results in total cases sampled. This can easily evaluate a binary classification problem. A rigorous metric is the precision score, which states what proportion of predicted positives are correct. This measures how near the predicted results are to one another. While the recall score is handy by measuring the proportion of actual positives to correctly classified. This measure is useful in diagnosis classifications problems, as we want to catch the disease even if we are not sure. The F1 score is the mean of precision and recall, and a score of zero indicates a poor model.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

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

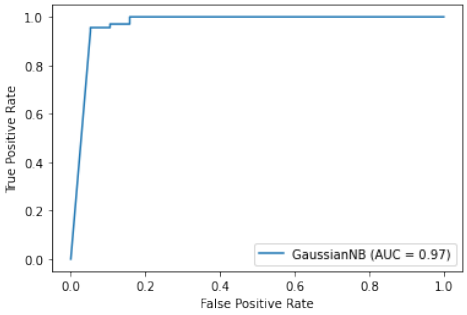
$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

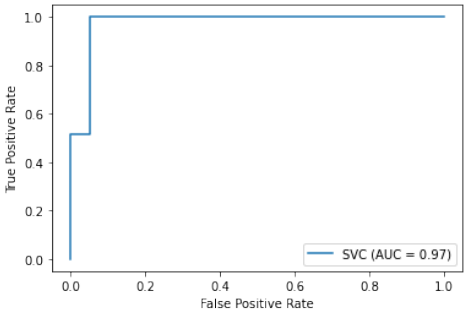
3 Results

Data was normalized and split into training set n = 348, validation set n = 44, and test set n = 43; all groups contained 33376 gene counts. Training set correlation was calculated of 1237 parameters at 0.3 threshold. The top 500 features in the training set were determined by RFE and a decision tree classifier. The data was further reduced by selecting the top 200 features by random forest classifier. With these features the best hyperparameters were selected by the process described in 2.4.1. The behavior of the neural network was assessed by modeling with a simple sequential deep neural network. Further assessment by cross validation confirmed the best hyperparameters. The best optimizer to adjust the weights between neurons was Adam. The best earning rate: 0.01, hidden layers: (20,), activation: ReLu. The performance of the neural network is high with an 0.95 AUC (Figure 2d) and high rate of values being accurately predicted (Figure 2d). The neural network performed well on the validated dataset and had similar accuracy and model loss to the training dataset. When performance accuracy was evaluated on the test dataset, all evaluation metrics were ¡ 0.95 (Table 3).

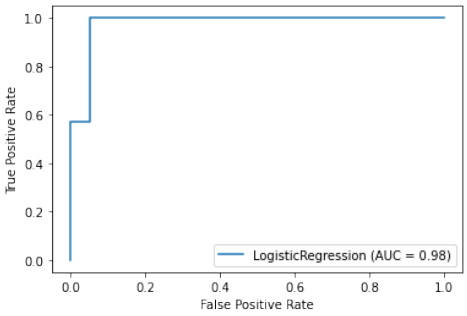
With the top hyperparameters verified in the neural network, two classifiers performance was analyzed. The Naive Bayes' classifier performed well with an accuracy of 0.90 and an AUC of 0.94.



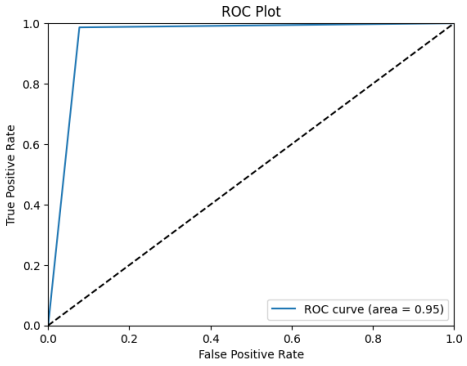
(a) ROC of Naive Bayes Classifier on Test Data



(b) ROC of Support Vector Machine Classifier on Test Data



(c) ROC of Logistic Regression Classifier on Test Data



(d) ROC of the Neural Network model on Test Data

Figure 2: ROC plots of Naive Bayes, Support Vector, Logistic Regression Classifiers and the Neural Network model on Test Data

	AUC	Accuracy	Precision	Recall	F1 score
Naive Bayes	0.97	0.95	0.98	0.96	0.97
Support Vector	0.97	0.98	0.98	0.99	0.99
Logistic Regression	0.98	0.98	0.97	1.0	0.98
Neural Network	0.95	0.96	0.99	0.97	0.98

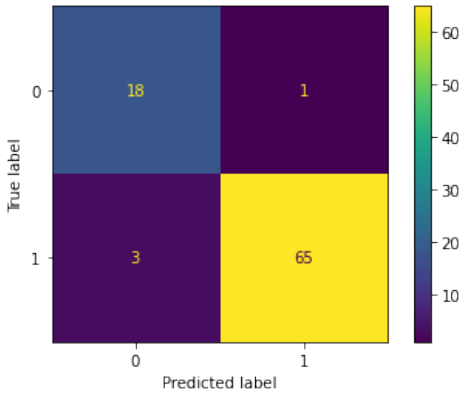
Table 3: Evaluation metrics on Naive Bayes, Support Vector and Logistic Regression Classifier on Test Data

SVM model performed slightly better with an accuracy of 0.93,with an AUC of 0.84. Additionally logistic regression was tested and had an accuracy of 0.97 and an AUC of 0.98. Both SVM and Naive Bayes' has an accuracy of 0.97 whereas Logistic Regression has an accuracy of 0.98. Therefore, Logistic Regression is a strong predictor of MSI status in colorectal cancer. The other parameters such as precision, recall, F1 score of every model has been observed. The precision of the SVM was 0.98, recall was 0.99 and F1 score was 0.99. The precision,recall and F1 score of the Naive Bayes was observed to be 0.98,0.96 and 0.97. Lastly, the model Logistic Regression when tested using the test set provided a precision, recall, F1 score of 0.97, 1.0 and 0.98.

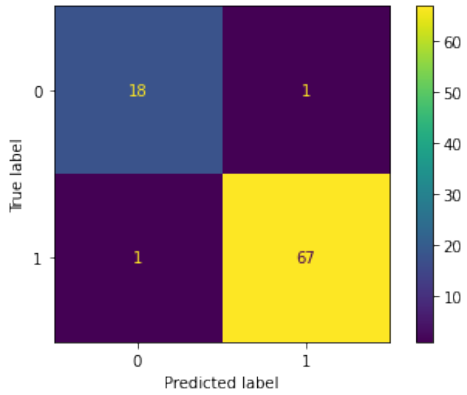
It is clear from Table 3 that Logistic Regression is the best model which has been able to classify our class labels.

4 Discussion

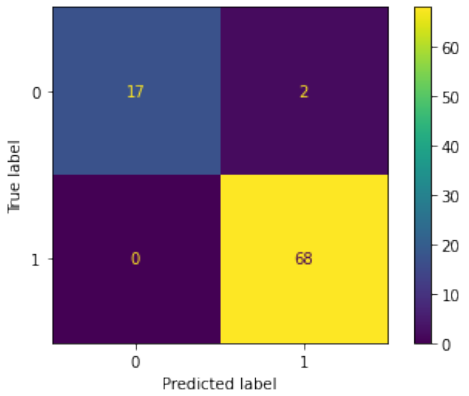
MSI status in colorectal cancer data has high dimensionality, therefore selecting a good model is a challenge. By testing hyperparameters with deep learning neural network modeling our analysis revealed high accuracy in all models tested. Logistic regression, the high accuracy, precision, recall,



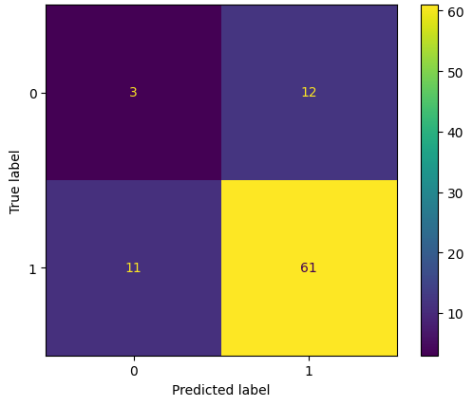
(a) Confusion Matrix of Naive Bayes Classifier on Test Data



(b) Confusion Matrix of Support Vector Machine Classifier on Test Data



(c) Confusion Matrix of Logistic Regression Classifier on Test Data



(d) Confusion Matrix of the Neural Network Model on Test Data

Figure 3: Confusion Matrix plots of Naive Bayes, Support Vector and Logistic Regression Classifier on Test Data

and F1 score of Naive Bayes, neural network, and SVM classifiers suggest that they are useful for predicting MSI status in colorectal cancer. However, the fact that logistic regression outperformed these classifiers in accuracy, precision, recall, F1 score and AUC score indicates that it may be more appropriate for this task. Taken together, these results demonstrate that all the models are strong predictors of MSI status in colorectal cancer data and are useful in identifying patients who may benefit from MSI-directed therapy in the clinical setting.

The fact that logistic regression is a simple but powerful algorithm that can handle binary classification problems such as MSI spatial classification of colorectal data can help it outperform other classifiers. If the classes are well separated, logistic regression can be very powerful because it finds the line of best fit in the feature space that divides the two classes. Furthermore, logistic regression is a linear classifier whereas classifiers such as SVM are non linear, which means logistic regression works well when the relationship between the input and output functions is linear or can be approximated by a linear function. If the relationship between the inputs and the output is relatively simple, this is useful for some data sets. In this case, the classification of MSI status given 200 most important features. On the other hand, SVM classifiers, neural networks and naive Bayes are more sophisticated and can handle more complex relationships between inputs and outputs. However, this increased complexity can lead to insufficient generalization and an excess of new data. Naive Bayes classifiers make the assumption that the input features are conditionally independent given the class label. This assumption may not hold in all cases.

5 Conclusion

In general, the choice of classifier is determined by the requirements of the particular problem, and each algorithm has distinct advantages and disadvantages. Due to its simplicity and ability to handle linear relationships between input functions and outputs, logistic regression can perform better than other classifiers in this scenario. However, it is always important to evaluate the performance of several classifiers on the same dataset and choose one that meets the requirements of a given problem

6 References

Fontana E, Eason K, Cervantes A, Salazar R, Sadanandam A. Context matters-consensus molecular subtypes of colorectal cancer as biomarkers for clinical trials. Ann Oncol. 2019 Apr 1;30(4):520-527. doi: 10.1093/annonc/mdz052. PMID: 30796810; PMCID: PMC6503627.

Dekker E, Tanis PJ, Vleugels JLA, Kasi PM, Wallace MB. Colorectal cancer. Lancet. 2019;394(10207):1467-1480. doi:10.1016/S0140-6736(19)32319-0

Lakbir S, Lahoz S, Cuatrecasas M, et al. Tumour break load is a biologically relevant feature of genomic instability with prognostic value in colorectal cancer. Eur J Cancer. 2022;177:94-102. doi:10.1016/j.ejca.2022.09.034

Kawakami H, Zaanan A, Sinicrope FA. Microsatellite Instability Testing and Its Role in the Management of Colorectal Cancer. Curr Treat Options Oncol. 2015;16(7). doi:10.1007/s11864-015-0348-2

Li Y, Ma Y, Wu Z, et al. Tumor Mutational Burden Predicting the Efficacy of Immune Checkpoint Inhibitors in Colorectal Cancer: A Systematic Review and Meta-Analysis. Front Immunol. 2021;12(September). doi:10.3389/fimmu.2021.751407

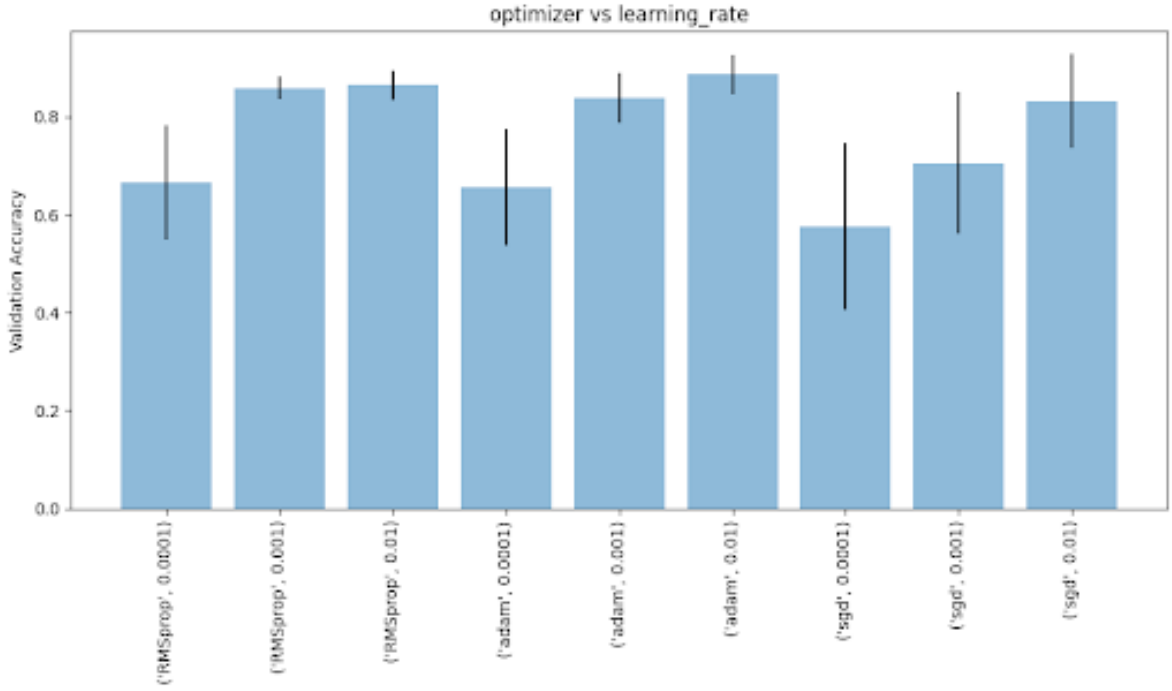
Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001).
https://doi-org.vu-nl.idm.oclc.org/10.1023/A:1010933404324

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. Machine learning, 46, 389-422.

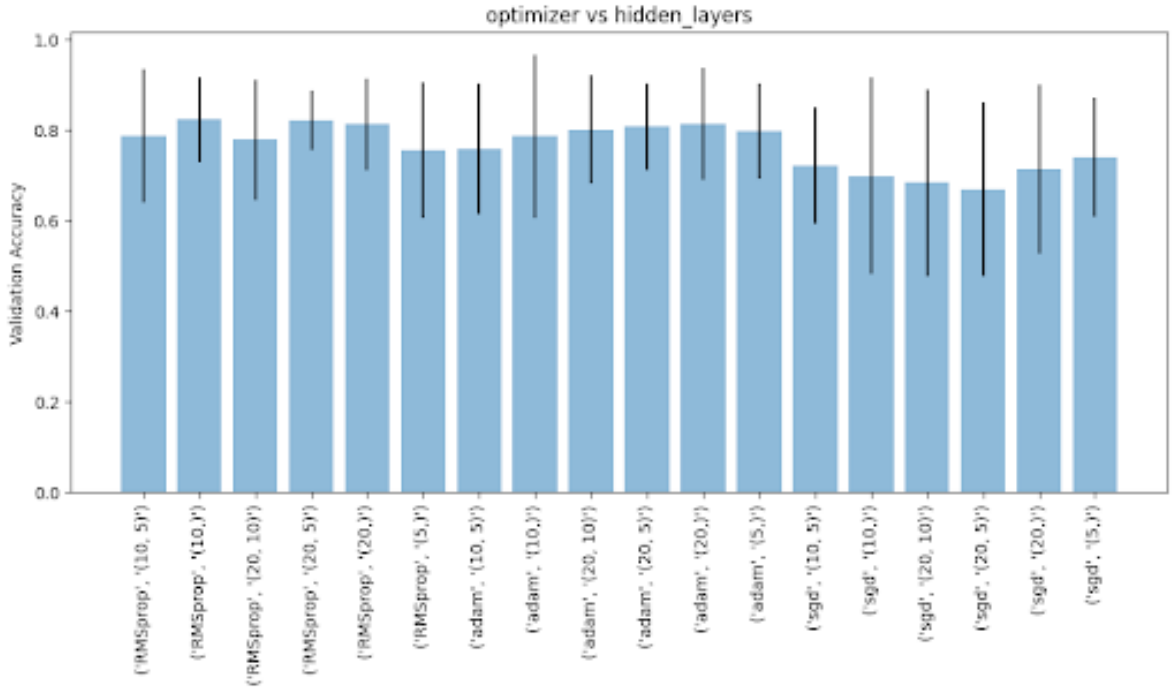
Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2016). Tensorflow:

Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

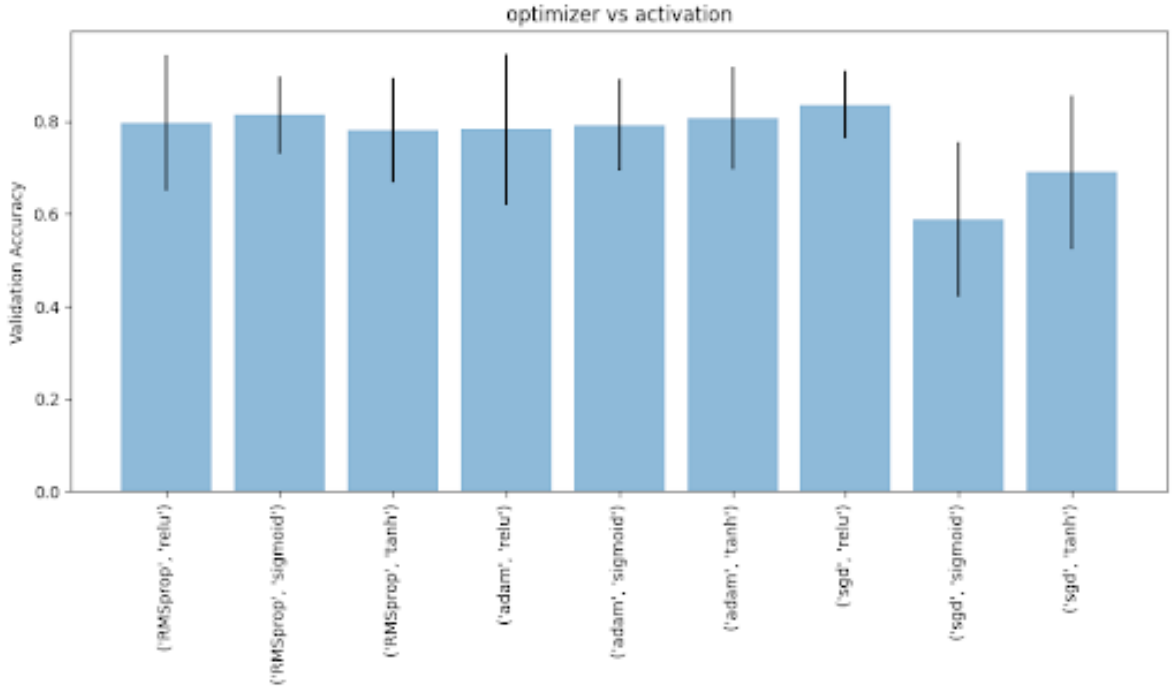
1 Supplementary



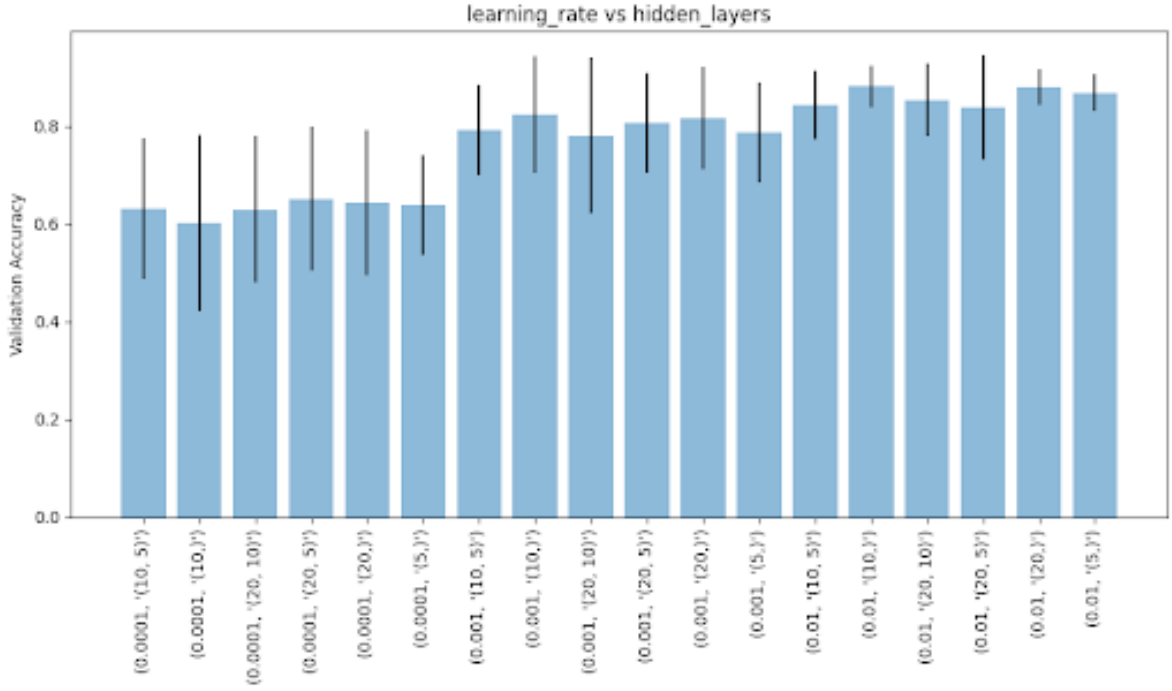
(a) Optimizer vs learning rate



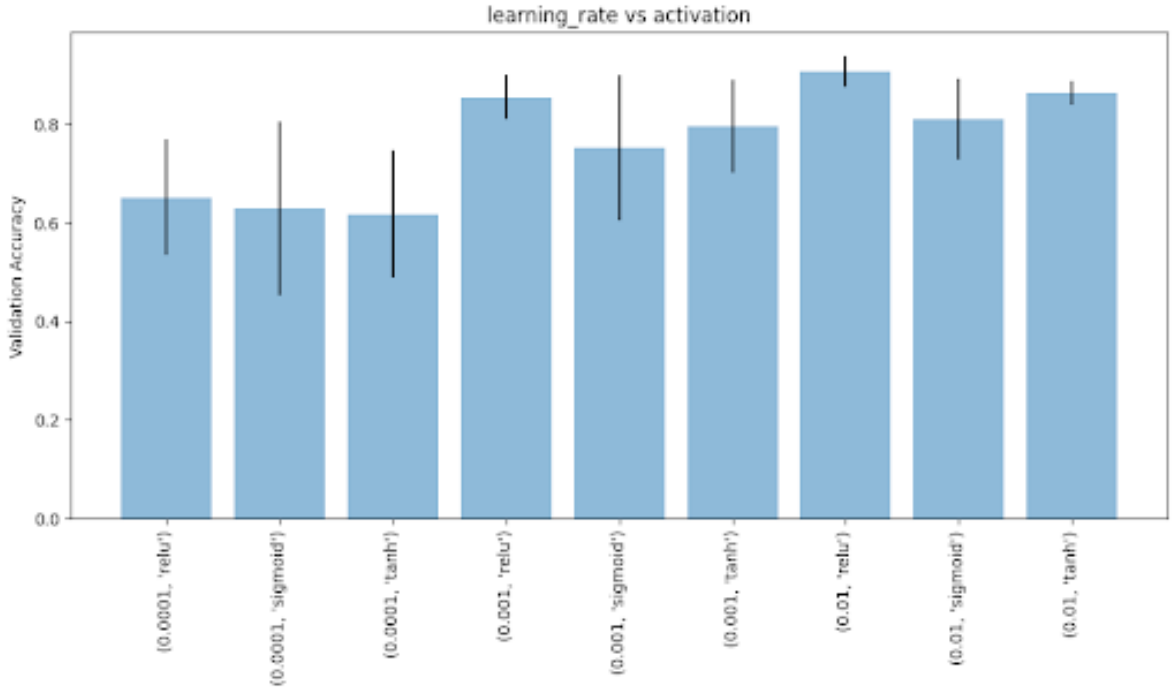
(b) Optimizer vs Hidden Layers



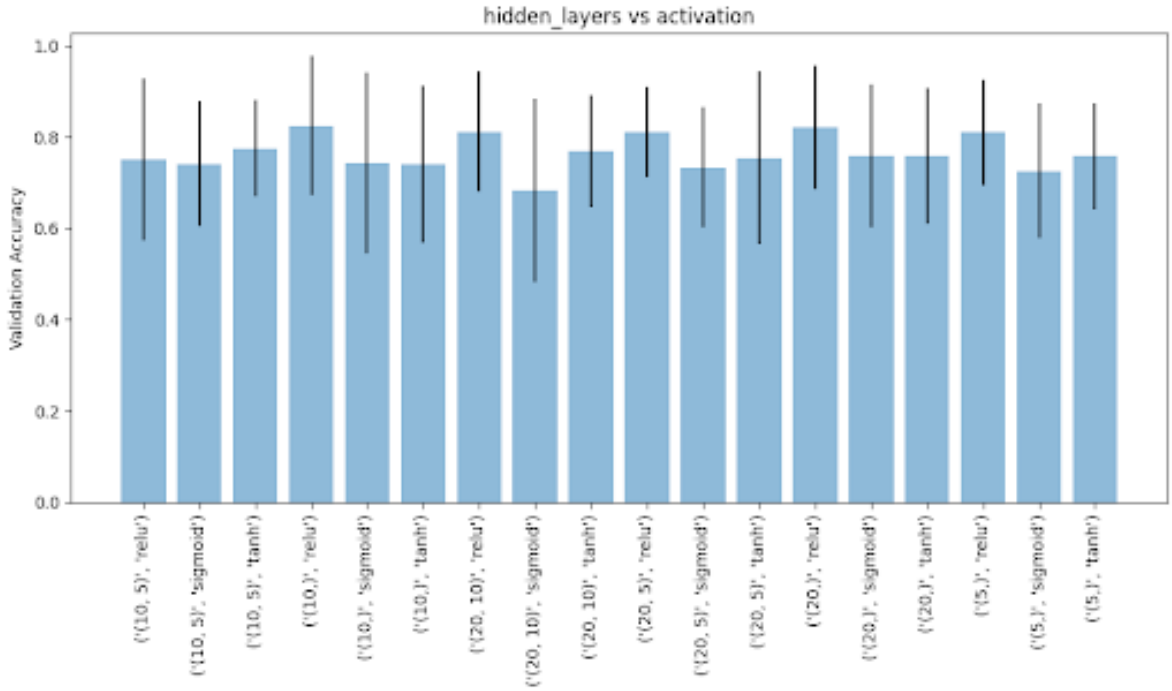
(a) Optimizer vs Activation



(b) Learning rate vs hidden layers



(a) Learning rate vs activation



(b) Hidden layers vs activation

Figure 3: Plots comparing mean and standard deviation of validation accuracy vs the different combinations of hyperparameters