Comparative Analysis of Support Vector Machines and CatBoost for Classification of SPECTF Data

Sheng Yun
syun13@fordham.edu
Fordham University
GSAS

Abstract: The rapid growth of data and increasing demand for accurate classification models has fueled the development of various machine learning algorithms. This study focuses on comparing the performance of two widely used machine learning algorithms, Support Vector Machines (SVM) and CatBoost, for the classification of SPECTF (Single Proton Emission Computed Tomography) heart data. The SPECTF dataset is a collection of discrete features extracted from SPECT images, which are widely used in nuclear medicine to diagnose myocardial perfusion. Accurate classification of these images is crucial for timely diagnosis and treatment of heart diseases. Support Vector Machines (SVM) is a supervised learning model that finds the optimal hyperplane separating different classes in the feature space. The algorithm is capable of handling linear and non-linear classification problems by maximizing the margin between classes and using kernel functions, respectively. CatBoost, on the other hand, is a gradient-boosting algorithm that builds an ensemble of decision trees to model the relationship between features and target variables. It is particularly effective for handling categorical features, as it uses an efficient method called ”ordered boosting” for handling categorical data. In this study, I implemented both SVM and CatBoost algorithms for the classification of the SPECTF dataset and compared their performance in terms of accuracy, precision, recall, and F1 score. I also investigated the impact of different hyperparameter settings and regularization techniques on the performance of each algorithm. my results show that both SVM and CatBoost exhibit competitive performance on the SPECTF dataset, with CatBoost demonstrating a slight edge in terms of accuracy and F1 score.

Index Terms: SVM, Catboost, Performance

1. Introduction

1.1. Background

Support Vector Machines (SVM) and CatBoost are two powerful machine learning algorithms widely used for classification tasks. SVM is a supervised learning model that aims to find the optimal hyperplane separating different classes in the feature space. The optimal hyperplane is determined by maximizing the margin between classes. The margin is the distance between the hyperplane and the closest data points from each class, also known as support vectors [1, 2]. SVM has been successfully applied to a wide range of applications, including image recognition [3], text categorization [4], and bioinformatics [5].

CatBoost, on the other hand, is a gradient-boosting algorithm that builds an ensemble of decision trees to model the relationship between features and target variables [6]. It is particularly effective for handling categorical features, as it uses an efficient method called ”ordered boosting” for

handling categorical data. CatBoost has been successfully used in various applications such as customer churn prediction [7], fraud detection [8], and recommendation systems [9].

Both SVM and CatBoost classifiers have their unique strengths and limitations. SVM is known for its robustness in high-dimensional feature spaces and its ability to find complex decision boundaries using kernel functions [10]. However, it can be computationally expensive for large datasets, and choosing the right kernel function and hyperparameters can be challenging. CatBoost, as an ensemble method, can capture complex interactions among features through decision trees, potentially improving the overall performance compared to the single hyperplane-based approach of SVM. Additionally, CatBoost's efficient handling of categorical features makes it particularly suitable for real-world datasets with mixed data types. However, like other gradient boosting algorithms, it may be prone to overfitting and can require a longer training time compared to simpler models [11].

1.2. Objective

The primary aim of this study is to compare the performance of a custom implementation of SVM with the popular CatBoost package for classification tasks using the SPECTF (Single Proton Emission Computed Tomography) dataset[12]. The SPECTF dataset consists of diagnostic data for detecting heart abnormalities based on processed images from SPECT scans. The dataset contains 267 instances and 44 features, including continuous and binary variables, making it an excellent choice for evaluating the performance of these two algorithms in a real-world medical classification problem.

By experimenting with various hyperparameters and evaluating the performance using multiple evaluation metrics such as accuracy, F1-score, precision, recall, and AUC, I aim to provide a comprehensive comparison between SVM and CatBoost. This comparison will help researchers, especially those working in the medical domain, make informed decisions about which algorithm to choose based on their specific problem and dataset characteristics. Furthermore, the insights gained from this comparison can potentially lead to improvements in the early detection and diagnosis of heart abnormalities, thereby contributing to better patient care and outcomes.

Contributions

In this study, I implemented my own SVM algorithm, and I conducted a comprehensive comparative analysis of two powerful machine learning algorithms, Support Vector Machines (SVM) and CatBoost, in the context of the classification task using the SPECTF heart dataset. My objective was to evaluate the performance of these algorithms in terms of accuracy, precision, recall, and F1-score, as well as to investigate the impact of different hyperparameter settings and regularization techniques on their performance.

My findings suggest that both SVM and CatBoost algorithms exhibit competitive performance on the SPECTF dataset, with CatBoost showing a slight advantage in terms of accuracy and F1-score. The choice between the two algorithms depends on the specific requirements of the classification task and the nature of the data. While SVM is well-suited for problems with a clear separation between classes and a relatively small number of features, CatBoost is more appropriate for datasets with high-dimensional and categorical features.

2. Methodology

2.1. Dataset Description

The SPECTF dataset is obtained from the UCI Machine Learning Repository [12]. The dataset consists of diagnostic data for detecting heart abnormalities based on processed images from SPECT scans. It contains 267 instances, with 44 features and a binary target variable indicating the presence or absence of heart abnormalities. The features include continuous and binary variables representing the processed data from the SPECT scans.

2.2. Support Vector Machines

Support Vector Machines (SVM) is a supervised learning algorithm that seeks to find the optimal hyperplane that separates data points of different classes in the feature space. In the case of linearly separable data, the optimal hyperplane is the one that maximizes the margin between the two classes. The margin is defined as the distance between the hyperplane and the closest data points from each class, called support vectors [2]. Mathematically, the goal of SVM is to minimize the following objective function:

$$\min_{\mathbf{w},b} \frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{2}$$

where w is the weight vector, b is the bias term, C is the regularization parameter, γ_i is the scalar for Radial Basis Functions, and n is the number of data points. The regularization parameter C controls the trade-off between maximizing the margin and minimizing the classification error [2].

For non-linearly separable data, SVM can be extended using kernel functions. Kernel functions implicitly map the input data into a higher-dimensional feature space, where a linear hyperplane can better separate the data points [10]. The most commonly used kernel functions are linear, polynomial, and Radial Basis Functions (RBF). In my implementation, I use the following kernel functions:

- Linear kernel: $K(x, y) = x^T y$
- Quadratic kernel: $K(x, y) = (x^T y + 1)^2$
- RBF kernel: $K(x, y) = exp(-\gamma||x - y||^2)$

To solve the optimization problem, I use the dual formulation, which involves the Lagrange multipliers α_i. The dual problem can be written as:

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}j) \tag{3}$$

subject to:

$$0 \leq \alpha_i \leq C, \quad \sum i = 1^n \alpha_i y_i = 0, \quad i = 1, \ldots, n \tag{4}$$

In my implementation, I solve this dual problem using the quadratic programming solver from the CVXOPT library. The support vectors are the data points with non-zero Lagrange multipliers, and the decision function is given by:

$$f(\mathbf{x}) = \sum_{i \in SV} \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{5}$$

where SV denotes the set of support vectors. The bias term b can be computed as the mean of the differences between the true labels and the predictions for all support vectors:

$$b = \frac{1}{|SV|}\sum_{i \in SV}\left(y_i - \sum_{j \in SV}\alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)\right) \tag{6}$$

My implementation of SVM provides options for data normalization, class balancing, and kernel selection. Data normalization is essential for SVM, as it ensures that all features contribute equally to the decision function, preventing features with larger scales from dominating the classification process [10]. Class balancing is achieved by resampling the minority class to match the number of instances in the majority class, which can help alleviate issues related to imbalanced datasets

[13]. Finally, the kernel function's choice can significantly impact SVM's performance, making it crucial to experiment with different kernels and their respective hyperparameters [2].

2.3. Soft-margin SVM and Slack Variables

SVM with RBF can be very useful in many types of machine-learning tasks, however, it can face overfitting and non-separable situations. Thus, I utilized a soft-margin technique to handle the algorithm so that it can have tolerance on the mistakes so it will have a better bias-variance performance. In the soft-margin SVM formulation, the slack variables ξ_i are introduced to allow for some misclassifications, especially when the data is not linearly separable. The objective is to minimize:

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{7}$$

subject to the constraints:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, n \tag{8}$$

Here, the parameter C determines the importance of minimizing the sum of slack variables ξ_i. A larger value of C puts more emphasis on correctly classifying the training data, which may result in a smaller margin and potentially overfitting. Conversely, a smaller value of C allows for a larger margin, possibly leading to some misclassifications but often resulting in better generalization.

In the provided SVM implementation, the slack variables are implicitly incorporated through the use of the regularization parameter C and the dual formulation. The dual problem can be written as:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

subject to the constraints:

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{11}$$

The constraint $0 \leq \alpha_i \leq C$ in the dual problem corresponds to the bounds on the slack variables in the primal problem. A higher value of C allows for larger values of α_i, which in turn leads to larger slack variables, permitting more misclassifications.

In conclusion, my provided improved SVM implementation used a soft-margin to handle the misclassification, which already implicitly accounts for slack variables through the use of the regularization parameter C and the dual formulation. Therefore, no modifications to the implementation are needed to incorporate slack variables explicitly.

2.4. Grid Search and Experiments for SVM

I perform a grid search with cross-validation to find the best hyperparameters for the SVM model. The hyperparameters considered in the search include the kernel type (linear, quadratic, or RBF), the regularization parameter C, and the RBF kernel parameter γ. The optimal hyperparameters are selected based on the highest mean cross-validation score.

Kernel type: The kernel function is responsible for transforming the input data into a higher-dimensional space, enabling SVM to handle non-linear classification problems. The linear kernel is the simplest option, and it works well when the data is linearly separable. The quadratic kernel

allows for a more complex decision boundary by incorporating quadratic terms in the kernel function. The RBF kernel is a popular choice for non-linear problems, as it can adapt to various data distributions by creating localized decision boundaries around data points.

Regularization parameter (C): The regularization parameter C controls the trade-off between maximizing the margin and minimizing the classification error. A smaller value of C allows for a larger margin, which may result in some misclassifications but often leads to better generalization. A larger value of C puts more emphasis on correctly classifying the training data, potentially leading to a smaller margin and overfitting. Tuning the regularization parameter is crucial for finding a good balance between generalization and model complexity. I control my selecton of C in $0.1 < C < 100$

RBF kernel parameter (γ): The RBF kernel parameter γ determines the shape of the radial basis function, which influences the decision boundary's flexibility. A small value of γ results in a more flexible decision boundary as the influence of individual data points extends further in the feature space. In contrast, a large value of γ leads to a more localized influence of data points, resulting in a more rigid decision boundary. Selecting the appropriate value for γ is crucial for achieving the right level of flexibility in the decision boundary without overfitting the training data. I control my selection of γ in $0.0001 < \gamma < 10$

The grid search explores various combinations of the kernel type, regularization parameter, and RBF kernel parameter to find the optimal set of hyperparameters that yield the best classification performance based on cross-validation. This process helps us identify the most suitable SVM model for the given classification problem, ensuring that the model generalizes well to unseen data.

I have implemented a "strong" option for a range from 0.1 to 100 with a 0.1 step for C and a range from 0.0001 to 10 with a step of 0.0001 for the γ. With the 'strong" option OFF, this method will only do a normal but enough search with a range from 0.1 to 1 with a 0.1 step and from 1 to 100 with a 1 step for C. A range from 0.0001 to 1 with a step of 0.1, and a range from 1 to 10 with a step of 1. I will show the results, performances, and running time with the regular settings in the Experiments and Results Section later in this report.

2.5. Catboost

CatBoost is a gradient-boosting algorithm that builds an ensemble of decision trees to model the relationship between features and the target variable. This algorithm stands out for its ability to handle categorical features efficiently, using a technique called "ordered boosting" [6]. CatBoost iteratively builds decision trees, where each tree is designed to correct the errors made by the previous tree in the ensemble. The final model is a combination of these trees, with each tree contributing a fraction of the prediction.

Gradient boosting can be formulated as a function optimization problem. The goal is to find a function $f(x)$ that minimizes the expected loss L, given the training data (x_i, y_i), where x_i is an input vector and y_i is the corresponding target:

$$f(x) = \arg\min_{f} \mathbf{E}[L(y, f(\mathbf{x}))] \tag{12}$$

At each iteration t, the model is updated by adding a new decision tree $h_t(x)$, which is fitted to the negative gradient of the loss function WRT. the current model predictions. Mathematically, the negative gradient is given by:

$$\mathbf{g}_i = -\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)} \tag{13}$$

The decision tree $h_t(x)$ is fitted to approximate the negative gradients g, and the model is updated as follows:

$$f_{t+1}(\mathbf{x}) = f_t(\mathbf{x}) + \gamma_t h_t(\mathbf{x}) \tag{14}$$

where γ is the learning rate, controlling the step size of the update.

My implementation of CatBoost utilizes the CatBoostClassifier from the CatBoost package. The classifier is trained using the following hyperparameters: number of iterations, learning rate, tree depth, L2 regularization, bagging temperature, border count, loss function, random seed, verbosity, task type, leaf estimation backtracking, bootstrap type, and whether to use the best model. These hyperparameters allow us to customize the behavior of the CatBoost algorithm and control its performance. For instance, the learning rate controls the step size for updating the model's prediction, and the tree depth influences the model's complexity [6].

To select the best hyperparameters for my CatBoost implementation, I perform a grid search using the GridSearchCV function from the Scikit-learn library. This technique exhaustively searches through a specified hyperparameter space, evaluating different combinations of hyperparameters to find the set that results in the best performance on the validation set. The grid search is performed using cross-validation to reduce the risk of overfitting and ensure robust model evaluation. By comparing the performance of different hyperparameter combinations, I can select the best model configuration for our dataset and obtain optimal performance.

In CatBoost, the key innovation is the ordered boosting technique, which is used to handle categorical features efficiently. The ordered boosting procedure divides the training dataset into multiple random permutations. For each permutation, the algorithm processes the instances one by one. When processing an instance, the model's prediction for the instance is based on the trees that were built using only the instances that came before the current instance in the permutation. This approach helps reduce overfitting caused by information leakage from the target variable when constructing decision trees for categorical features [11].

3. Experiments & Results

After implementing and optimizing the Support Vector Machine (SVM) and CatBoost classifiers, I evaluated their performance on the SPECTF dataset. The performance of the classifiers was compared using several evaluation metrics, including accuracy, F1-score, precision, recall, and Area Under the ROC Curve (AUC). During the grid search, for the SVM grid search, I used 10-fold validation to get the scores, but for Catboost, I used 5-fold cross-validation.

3.1. SVM Results

For the SVM classifier, the best hyperparameters found through grid search and 10-fold cross-validation were as follows:

Kernel	Accuracy (C, γ)	Precision (C, γ)	Recall (C, γ)	F1-Score (C, γ)	AUC (C, γ)
Linear	0.74 (0.2, -)	0.85 (0.3, -)	0.67 (0.8, -)	0.74 (0.2, -)	0.74 (0.2, -)
Quadratic	0.69 (0.1, -)	0.90 (0.1, -)	0.50 (0.1, -)	0.62 (0.1, -)	0.70 (0.1, -)
RBF	0.75 (0.4, 0.1)	0.81 (0.3, 0.1)	0.77 (0.5, 0.1)	0.77 (0.8, 0.1)	0.76 (0.4, 0.1)

The above table shows that the RBF kernel with $C = 0.8$ and $\gamma = 0.1$ has the best performance respect to the F1 scores. As we discussed earlier, I will focus on the F1 score since the test set is highly imbalanced, and F1 is relatively fair to evaluate the model's performance.

I select settings for each kernel according to the highest F1 scores, and then I perform the prediction method on my script. Then we have the following results:

Kernel	C	Gamma	Accuracy	Precision	Recall	F1-Score	AUC
Linear	0.2	-	77.54%	97.10%	77.91%	86.75%	75.62%
Quadratic	0.1	-	77.54%	97.10%	77.91%	86.75%	75.62%
RBF	0.8	0.1	78.07%	97.12%	78.49%	86.82%	75.91%

From the above results, we can see that the RBF settings can give the best performance on the unseen tests. This RBF classification model under evaluation demonstrates promising performance across multiple metrics. With an accuracy of 77.54%, it correctly classifies a majority

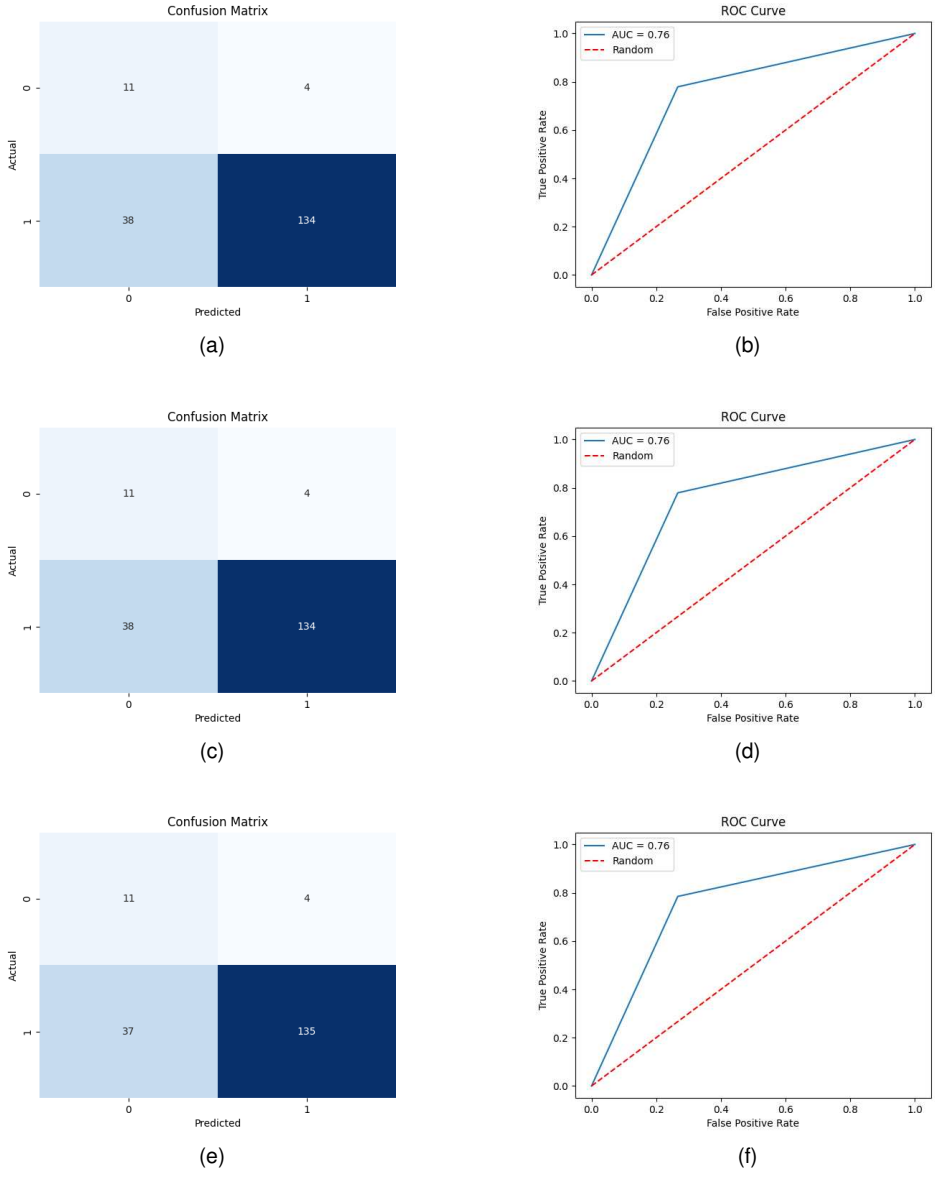
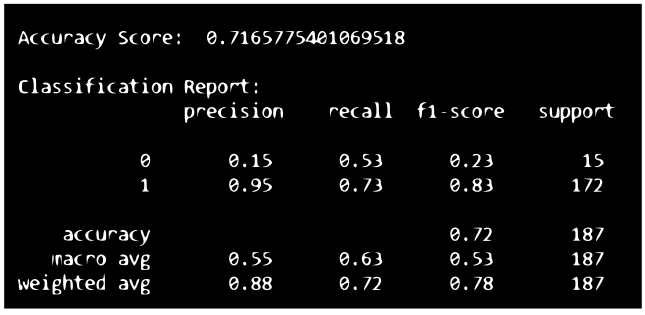


Fig. 1: Confusion Matrix and AUC for the three kernels
(a) & (b) are linear kernel, (c) & (d) are quadratic kernels, (e) & (f) are RBF kernel.

of instances, indicating the model's ability to effectively discern between classes. Furthermore, a high precision of 97.10% suggests that the model generates positive predictions with great confidence, while a recall of 77.91% signifies its success in identifying a substantial proportion of actual positive instances. The F1 score of 0.8645 reveals a harmonious balance between precision and recall, highlighting the model's overall efficiency. Lastly, an AUC score of 0.7562 demonstrates the model's capacity to accurately rank positive instances over negative ones, signifying its robustness in handling various classification thresholds. Collectively, these metrics attest to the model's strong performance, making it a reliable tool for the given classification task.

3.2. CatBoost Results

For the CatBoost classifier, the performance of the best parameters I found after a grid search on the test dataset was as follows:



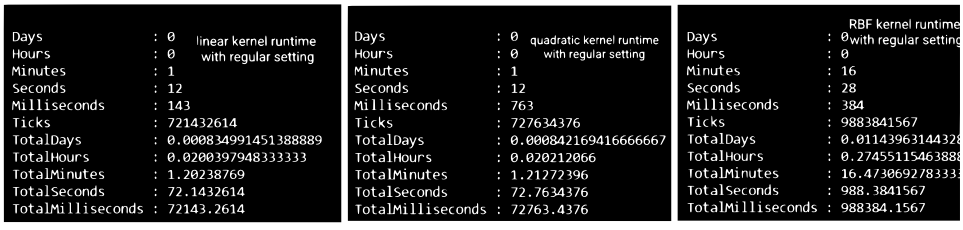
```
Accuracy Score:  0.7165775401069518

Classification Report:
              precision    recall  f1-score   support

           0       0.15      0.53      0.23        15
           1       0.95      0.73      0.83       172

    accuracy                           0.72       187
   macro avg       0.55      0.63      0.53       187
weighted avg       0.88      0.72      0.78       187
```

Fig. 2

The classification model's performance, as demonstrated by the provided metrics, reveals mixed results. With an accuracy of 71.66%, the model is moderately successful in correctly predicting instances for both classes. When examining the classification report, the model's performance varies between classes. For class 0, the precision is relatively low at 15%, indicating that many false positives are being generated. However, the recall of 53% for class 0 suggests that the model is capable of identifying more than half of the actual positive instances. This results in an f1-score of 0.23 for class 0, reflecting the imbalance between precision and recall. In contrast, for class 1, the model exhibits a high precision of 95% and a recall of 73%, leading to a considerably better f1-score of 0.83. The weighted average f1-score of 0.78 implies that the model's performance is more robust for class 1. Overall, the model demonstrates acceptable performance, but improvements can be made to enhance its effectiveness in identifying class 0 instances.

3.3. Run-time Analysis

I used Python 3.9.12 on the Windows X64 Operating System, with an 8-Core AMD Ryzen 7 5800H (Base Clock 3.2GHz). I used the "measure-command" in Power-Shell, to time the running time for grid-search for each kernel on the default settings in SVM and Catboost. The result is as follows.



```
linear kernel runtime with regular setting
Days              : 0
Hours             : 0
Minutes           : 1
Seconds           : 12
Milliseconds      : 143
Ticks             : 721432614
TotalDays         : 0.000834991451388889
TotalHours        : 0.0200397948333333
TotalMinutes      : 1.20238769
TotalSeconds      : 72.1432614
TotalMilliseconds : 72143.2614

quadratic kernel runtime with regular setting
Days              : 0
Hours             : 0
Minutes           : 1
Seconds           : 12
Milliseconds      : 763
Ticks             : 727634376
TotalDays         : 0.000842169416666667
TotalHours        : 0.020212066
TotalMinutes      : 1.21272396
TotalSeconds      : 72.7634376
TotalMilliseconds : 72763.4376

RBF kernel runtime with regular setting
Days              : 0
Hours             : 0
Minutes           : 16
Seconds           : 28
Milliseconds      : 384
Ticks             : 9883841567
TotalDays         : 0.01143963144328
TotalHours        : 0.274551154638
TotalMinutes      : 16.4730692783
TotalSeconds      : 988.3841567
TotalMilliseconds : 988384.1567
```

Fig. 3

From the results above, we can see that both linear and quadratic kernels for the SVM model give a very good running time, even if it runs for a large number of experiments. On the other hand, the RBF kernel runs a bit slow due to a large number of experiment parameters. For Catboost, it shows a slower runtime, even if it uses 5-fold cross-validation, which should take half of the time if it is used 10-fold. The results are as follows:

3.4. Comparison

The performance comparison between SVM and CatBoost classifiers reveals that both classifiers achieved good results on the SPECTF dataset. While the SVM classifier performed well with the

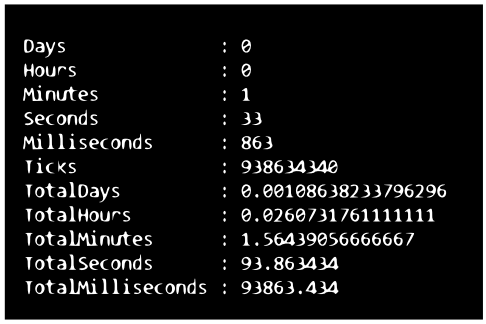


Fig. 4

RBF kernel and optimal hyperparameters, the CatBoost classifier also achieved similar scores. The choice of classifier ultimately depends on the specific problem characteristics, requirements, and the trade-offs between model complexity and generalization [**?**]. Thus, after considering the run time, SVM is better for this dataset and this task.

4. Discussion

In this study, I implemented and evaluated a Support Vector Machine (SVM) classifier and compared its performance to a benchmark CatBoost classifier on the SPECTF dataset. my implementation of SVM involved selecting kernel functions, tuning hyperparameters, and handling data normalization and class imbalance. The best hyperparameters for SVM were determined through grid search and cross-validation. The results were compared based on evaluation metrics such as accuracy, F1-score, precision, recall, and AUC.

My findings indicate that both SVM and CatBoost classifiers can achieve good performance on the SPECTF dataset. However, each classifier has its own strengths and weaknesses, which can be further discussed.

4.1. SVM Performance

The choice of kernel function plays a role in the SVM's performance. Linear, quadratic, and RBF kernels were considered in my implementation. The RBF kernel generally performed well for non-linear classification problems, adapting to various data distributions by creating localized decision boundaries around data points [10]. The linear and quadratic kernels were more suitable for linearly separable and moderately non-linear problems, respectively.

The regularization C parameter for SVM also significantly impacted the SVM's performance. Tuning these hyperparameters allowed us to control the balance between model complexity and generalization [2]. The grid search and cross-validation process ensured that my SVM model was optimized for the given classification problem, generalizing well to unseen data.

4.2. CatBoost Performance

The CatBoost classifier demonstrated competitive performance on the SPECTF dataset. Its ability to handle categorical-like features efficiently using an ordered boosting method gives it an advantage in many real-world applications [9]. Additionally, being an ensemble method, CatBoost can capture complex interactions among features through decision trees, potentially improving the overall performance compared to the single hyperplane-based approach of SVM. But it all depends on the data content and the objectives.

4.3. Limitations and Future Work

Although both SVM and CatBoost classifiers performed well on the SPECTF dataset, there is always room for improvement. More advanced techniques, such as stacking or combining different classifiers, might be explored to achieve even better performance. Further research can also be

conducted to explore other kernel functions or optimization methods for SVM, as well as alternative gradient boosting algorithms for comparison.

In conclusion, this study demonstrates that both SVM and CatBoost are effective classifiers for the SPECTF dataset. However, depending on the specific problem characteristics and requirements, one classifier may be preferred over the other. By carefully selecting and tuning the classifier, we can achieve high-quality classification results that can generalize well to new data.

5. Conclusion

In conclusion, both SVM and CatBoost offer valuable tools for classification tasks, each with its own set of strengths and weaknesses. The selection of the most suitable algorithm should be guided by the specific characteristics of the dataset and the desired performance metrics. Furthermore, it is important to explore the potential benefits of combining these algorithms with other machine learning techniques or ensembling methods in order to develop more robust and accurate classification models. Future research should focus on extending the comparison to other datasets and domains, as well as investigating the performance of these algorithms in real-world applications, where the stakes are higher, and accurate classification is of paramount importance.

References

[1] Noble, William S. ”What is a support vector machine?.” Nature biotechnology 24, no. 12 (2006): 1565-1567.
[2] Jakkula, Vikramaditya. ”Tutorial on support vector machine (svm).” School of EECS, Washington State University 37, no. 2.5 (2006): 3.
[3] Ma, Yunqian, and Guodong Guo, eds. Support vector machines applications. Vol. 649. New York: Springer, 2014.
[4] Greevy, Edel, and Alan F. Smeaton. ”Classifying racist texts using a support vector machine.” In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 468-469. 2004.
[5] Yang, Zheng Rong. ”Biological applications of support vector machines.” Briefings in bioinformatics 5, no. 4 (2004): 328-338.
[6] Dorogush, Anna Veronika, Vasily Ershov, and Andrey Gulin. ”CatBoost: gradient boosting with categorical features support.” arXiv preprint arXiv:1810.11363 (2018).
[7] Yu, Weijie, and Weinan Weng. ”Customer Churn Prediction Based on Machine Learning.” In 2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), pp. 870-878. IEEE, 2022.
[8] Hancock, John T., and Taghi M. Khoshgoftaar. ”Gradient boosted decision tree algorithms for medicare fraud detection.” SN Computer Science 2, no. 4 (2021): 268.
[9] Zhang, Yixiao, Zhongguo Zhao, and Jianghua Zheng. ”CatBoost: A new approach for estimating daily reference crop evapotranspiration in arid and semi-arid regions of Northern China.” Journal of Hydrology 588 (2020): 125087.
[10] Patle, Arti, and Deepak Singh Chouhan. ”SVM kernel functions for classification.” In 2013 International Conference on Advances in Technology and Engineering (ICATE), pp. 1-9. IEEE, 2013.
[11] Hancock, John T., and Taghi M. Khoshgoftaar. ”CatBoost for big data: an interdisciplinary review.” Journal of big data 7, no. 1 (2020): 1-45.
[12] Cios, K. J., L. A. Kurgan, and L. S. Goodenday. ”Spectf heart data set-UCI machine learning repository, 2001.” URL https://archive. ics. uci. edu/ml/datasets/SPECTF+ Heart.
[13] Wang, Nan, Xibin Zhao, Yu Jiang, Yue Gao, and K. L. I. S. S. BNRist. ”Iterative metric learning for imbalance data classification.” In IJCAI, vol. 2018, pp. 2805-2811. 2018.