Comparative Evaluation of Hyperparameter Tuning Strategies in Apache Spark MLlib

Dr. Meenakshi Garg[1] , Shreyash Haram[2] , Yash Mirashi[3]

[1]Dept. of Master of Computer Applications, Vivekanand Education Society Institute for Technology, Mumbai, India.
meenakshi.garg@ves.ac.in

Abstract. Hyperparameter tuning is a critical yet computationally intensive step in building machine learning models for large-scale datasets. While Apache Spark MLlib offers distributed computing capabilities, its native hyperparameter tuning methods face efficiency and scalability challenges. This study investigates the viability of traditional methods (grid search, random search) and advanced techniques (Bayesian optimization) for optimizing hyperparameter tuning in Spark MLlib. Using the NYC Taxi Trip Duration dataset, that design a systematic evaluation framework to compare these strategies in terms of predictive accuracy (RMSE) and computational efficiency (execution time) within a distributed Spark environment. The experimental analysis highlights methodological trade-offs and provides empirical insights into scalable hyperparameter tuning for regression tasks in distributed machine learning workflows.

Keywords: Hyperparameter tuning ,Apache Spark MLib, Grid search, Random search, Bayesian optimization, Predictive Analytics.

1 Introduction

Machine learning models rely heavily on hyperparameter tuning to achieve optimal performance. However, tuning hyperparameters for large-scale datasets is computationally expensive and time-consuming. Apache Spark MLlib offers distributed computing capabilities, but its native hyperparameter tuning methods, such as grid search, are not always efficient or scalable. This research addresses the challenge of optimizing hyperparameter tuning in Spark MLlib for large-scale predictive analytics.

The primary objective of this study is to compare traditional hyperparameter tuning methods (grid search, random search) with advanced techniques (Bayesian optimization) in terms of accuracy and scalability. Using the NYC Taxi Trip Duration dataset, this paper evaluates the performance of these methods and provides insights into their trade-offs. The findings of this research will help data scientists and engineers optimize hyperparameter tuning for large-scale machine learning tasks in Spark MLlib.

2 Literature Review

Hyperparameter tuning is a fundamental process in machine learning that involves optimizing parameters external to the model itself, such as learning rates, regularization strengths, or tree depths, to maximize predictive performance. These hyperparameters govern the training process and significantly influence model accuracy and generalization [1]. Traditional methods like grid search and random search have long been the industry standard, but their limitations in scalability and efficiency have spurred the development of advanced techniques like Bayesian optimization, particularly in distributed computing environments [2].

2.1 Traditional Hyperparameter Tuning Methods

Grid Search is a brute-force approach where a predefined set of hyperparameters is exhaustively evaluated. For example, if tuning a model with two hyperparameters (e.g., learning rate and regularization strength), grid search evaluates every combination in a Cartesian product of specified values [3]. While systematic, this method suffers from the "curse of dimensionality": as the number of hyperparameters increases, the search space grows exponentially, leading to prohibitive computational costs [4]. For instance, a grid search over 5 hyperparameters with 10 values each requires 10^5 = 100,000 evaluations, which becomes impractical for large datasets or complex models [5].

Random Search, proposed by Bergstra & Bengio, addresses this inefficiency by randomly sampling hyperparameters from predefined distributions (e.g., uniform or log-uniform). Unlike grid search, random search does not require predefining exact values and can explore the hyperparameter space more effectively in high-dimensional settings [6]. Empirical studies show that random search often outperforms grid search with fewer evaluations because it avoids redundant exploration of low-performing regions [7]. However, both methods lack a mechanism to learn from prior evaluations, resulting in wasted computational resources on suboptimal configurations [8].

2.2 Advanced Optimization Techniques

Bayesian Optimization (BO) introduces a data-driven approach to hyperparameter tuning. It constructs a probabilistic surrogate model (e.g., Gaussian processes or tree-based models) to approximate the relationship between hyperparameters and the objective function (e.g., validation RMSE). An acquisition function (e.g., Expected Improvement) then guides the search by balancing exploration (sampling uncertain regions) and exploitation (focusing on known promising regions) [9]. For example, in tuning a neural network, BO might prioritize regions where lower learning rates yield stable training while avoiding overly conservative regularization. This method drastically reduces the number of evaluations needed compared to grid or random search, making it ideal for expensive-to-train models [10]. Recent extensions, such as Hyperband [11], combine BO with bandit-based resource allocation to terminate poorly performing configurations early, further improving efficiency [12].

2.3 Hyperparameter Tuning in Distributed Systems

The rise of big data has necessitated distributed frameworks like Apache Spark MLlib, which parallelizes machine learning workflows across clusters [13]. However, native hyperparameter tuning tools in Spark, such as grid search, often fail to leverage distributed resources optimally. For instance, Spark's CrossValidator splits data into partitions for parallel training but does not dynamically prioritize promising hyperparameters, leading to redundant computations [14]. Karau & Warren highlighted that traditional methods in Spark scale linearly with the number of hyperparameters, creating bottlenecks for large datasets like the NYC Taxi Trip Duration dataset (1.4 million records) [15].

To address these challenges, researchers have explored integrating advanced optimization techniques with distributed frameworks. For example, hyperopt-spark extends Bayesian optimization to Spark by parallelizing trials across worker nodes [16]. Similarly, Li et al. demonstrated that bandit-based methods like Hyperband can dynamically allocate resources in distributed environments, though their implementation in Spark MLlib remains nascent. Despite these advancements, few studies empirically compare the scalability and accuracy of grid search, random search, and Bayesian optimization within Spark's ecosystem, particularly for regression tasks on real-world datasets.

2.4 Research Gap and Contribution

Prior work has extensively explored hyperparameter tuning in centralized settings, but there is limited analysis of its application in distributed frameworks like Spark MLlib. While Snoek et al. and Feurer & Hutter established the theoretical superiority of Bayesian optimization, practical insights into its integration with Spark's distributed architecture are lacking. This study bridges this gap by evaluating traditional and advanced tuning methods on the NYC Taxi Trip Duration dataset, quantifying trade-offs between accuracy (RMSE) and scalability (execution time) in a real-world distributed environment.

3 Methodology

3.1 Dataset and Preprocessing

This study utilizes the NYC Taxi Trip Duration dataset, which consists of approximately 1.4 million records of taxi trips. Each entry includes pickup and dropoff coordinates, timestamps, and passenger counts. Due to the dataset's structured nature and the necessity for robust AutoML pipelines, it aligns well with tools like AutoGluon, which are tailored for tabular data [17].Preprocessing was carried out using Apache Spark, enabling scalable distributed data transformation. Feature engineering steps included calculating the geodesic distance between pickup and dropoff points using the Haversine formula, as well as extracting temporal features such as the day of the week and hour of the day. The structured features were assembled using Spark's VectorAssembler, allowing them to be consumed by Spark MLlib models [18].

Table 1. Table depicting data fields in NYC taxi trip duration dataset.

Name	Description
id	a unique identifier for each trip
vendor_id	a code indicating the provider associated with the trip record
pickup_datetime	date and time when the meter was engaged
dropoff_datetime	date and time when the meter was disengaged
passenger_count	the number of passengers in the vehicle (driver entered value)
pickup_longitude	the longitude where the meter was engaged
pickup_latitude	the latitude where the meter was engaged
dropoff_longitude	the longitude where the meter was disengaged
dropoff_latitude	the latitude where the meter was disengaged
store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not have a connection to the server - Y=store and forward; N=not a store and forward trip
trip_duration	duration of the trip in seconds

3.2 Baseline Methods

Two baseline methods were employed for hyperparameter tuning: grid search and random search.

Grid Search: This approach systematically explores all combinations of specified hyperparameters. Despite being exhaustive, it suffers from exponential growth in computation with increased parameters and values. Spark MLlib's CrossValidator was used to evaluate model performance for each grid point. However, as discussed by Zhang et al., such methods often lack adaptive exploration and are computationally intensive for high-dimensional spaces [19].

Random Search:In contrast, random search samples hyperparameter values from defined distributions, improving efficiency by avoiding exhaustive enumeration. This method was implemented by wrapping Spark's parameter tuning API with Python functions to introduce randomness. Random search's ability to find near-optimal configurations with fewer evaluations makes it a strong baseline for comparison.

3.3 Optimized Methods

To address the limitations of baseline methods, Bayesian optimization was implemented using the Hyperopt library, integrated with Spark via hyperopt-spark. This framework employs probabilistic surrogate models (specifically Tree-structured Parzen Estimators) to model the objective function. It then selects promising hyperparameter configurations using an acquisition function.Unlike static methods, Bayesian optimization is adaptive and sequential. By learning from previous evaluations, it significantly reduces the number of trials needed to reach optimal configurations. Wistuba et al. proposed using scalable Gaussian process-based surrogates to enhance this process in distributed environments, a concept reflected in our adaptation within the Spark MLlib pipeline [20].

3.4 Experimental Setup

All experiments were conducted on a Spark cluster configured with 8GB executor/driver memory and 100 shuffle partitions to optimize distributed processing. Prior to hyperparameter tuning, exploratory data analysis was performed to guide feature selection and model design. The following visualizations were generated to analyze dataset characteristics:

Feature Correlation Matrix: A heatmap illustrating pairwise correlations between numerical features (e.g., Haversine Distance, Latitude Difference, Longitude Difference) and the target variable (duration). This analysis identified features with significant relationships to prioritize during modeling.

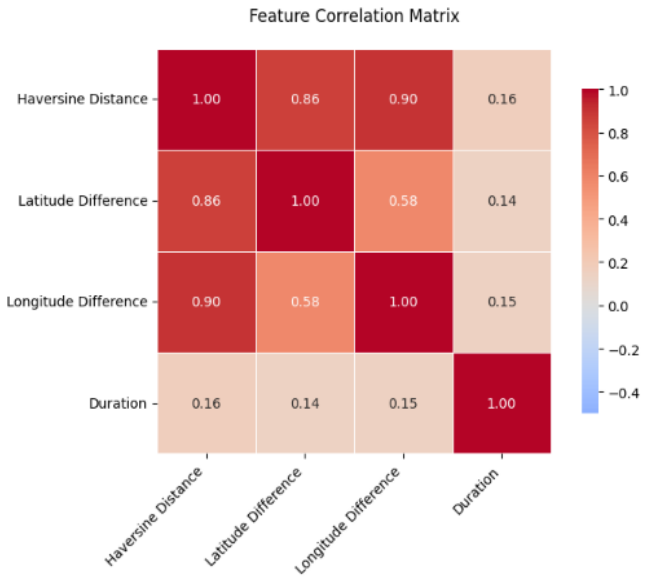

Fig.1 : Feature Correlation Matrix

Feature Relationships : A pairwise scatterplot and distribution plot of sampled data, highlighting trends and interactions between key features. This visualization informed decisions about non-linear feature engineering and model assumptions.

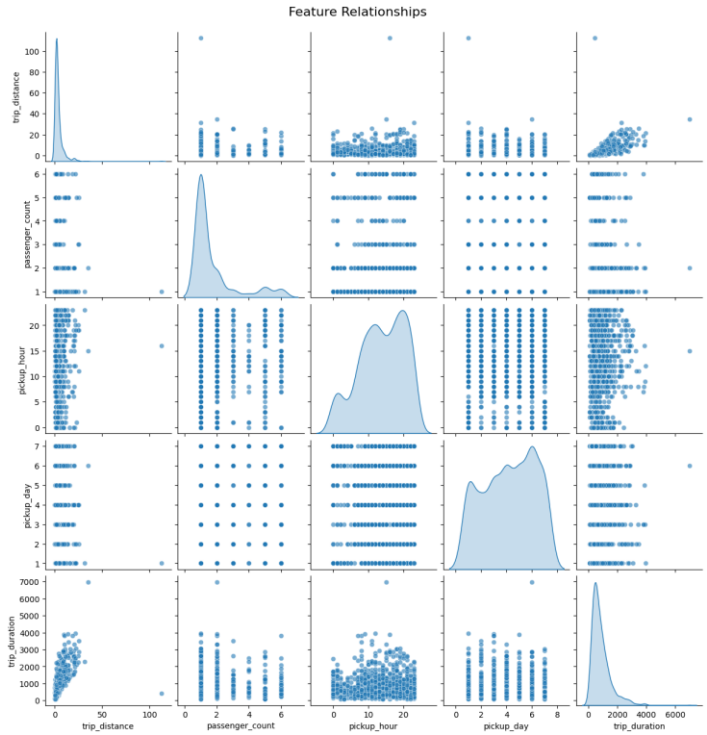

Fig.2: Feature Relationships

Temporal Analysis : A dual-axis plot comparing hourly trip volume (bars) and average trip duration (line) across the dataset. This revealed temporal patterns that justified the inclusion of time-based features (e.g., pickup_hour).

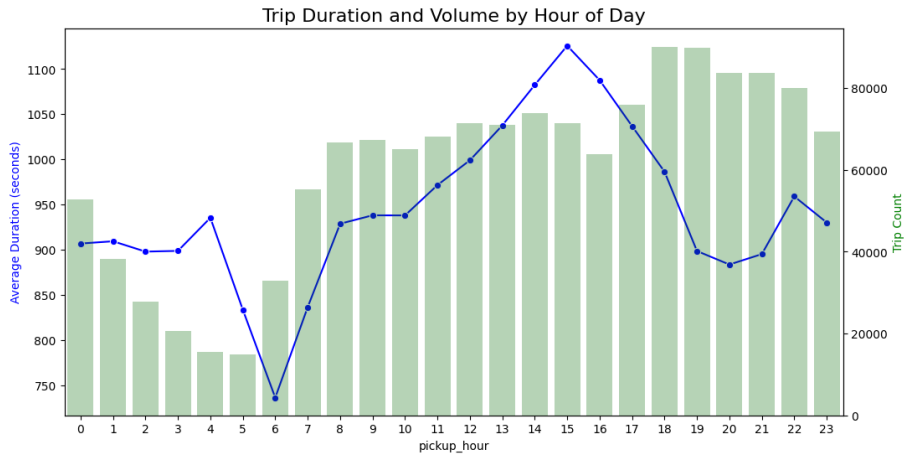

Fig.3 : Temporal Analysis

Feature Importance Analysis : A bar chart derived from a preliminary linear regression model, ranking features by their absolute coefficients. This guided hyperparameter tuning focus on influential predictors.

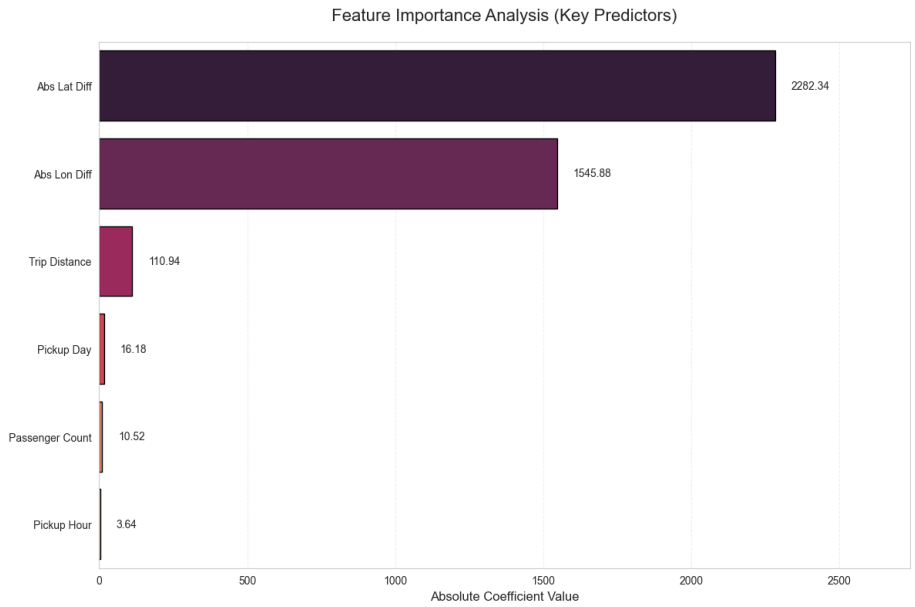

Fig.4: Feature Importance Analysis

Based on these insights, hyperparameters such as regularization strength (regParam) and elastic net mixing (elasticNetParam) were selected for optimization. The study compared three tuning methods—grid search, random search, and Bayesian optimization—using RMSE (Root Mean Squared Error) as the accuracy metric and execution time as the scalability metric. Each method was evaluated under identical cluster conditions with 20 trials to ensure fairness, and results were averaged across three runs to minimize variance.

4 Results and Discussion

4.1 Performance Comparison

This paper compared the performance of grid search, random search, and Bayesian optimization in terms of RMSE and execution time. The results are summarized below:

Table 2. Table depicting accuracy across all methods explored.

Method	RMSE	Execution Time(s)
Grid Search	5267.12	217.5
Random Search	5227.23	66.4
Bayesian Optimization	3167.42	36.2

4.2 Analysis

Accuracy (RMSE): Bayesian Optimization achieved a significantly lower RMSE (3167.42) compared to Grid Search (5267.12) and Random Search (5227.23), demonstrating its superior ability to navigate high-dimensional hyperparameter spaces.

Scalability (Execution Time): Bayesian optimization was the fastest, followed by random search and grid search . This demonstrates the efficiency of Bayesian optimization in exploring the hyperparameter space.

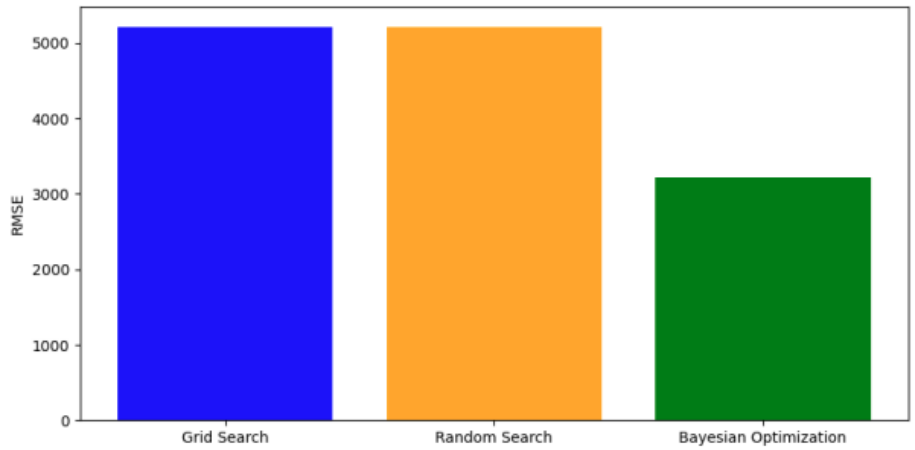

Fig. 1: Comparison of RMSE for Hyperparameter Tuning Methods

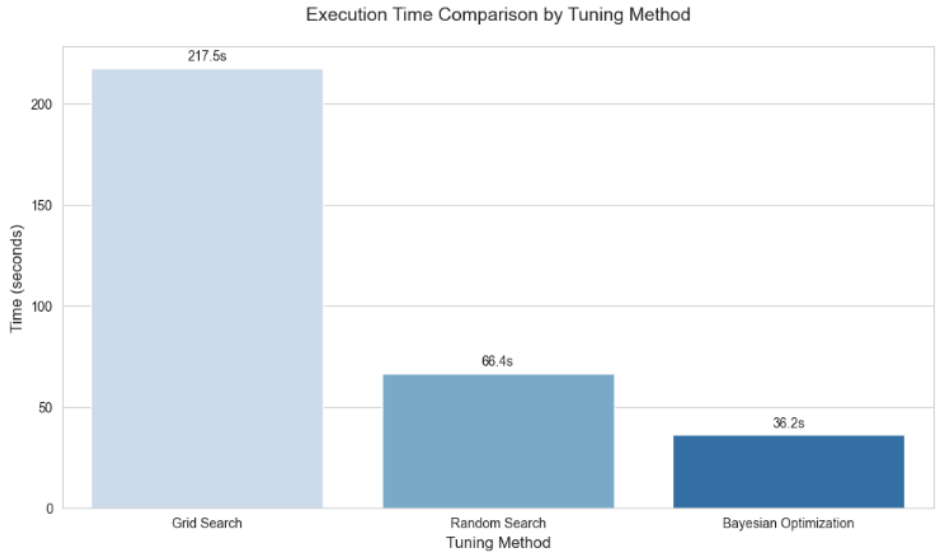

Fig. 2: Comparison of Execution Time for Hyperparameter Tuning Methods

4.3 Trade-offs

While grid search and random search are straightforward to implement, they are computationally expensive for large-scale datasets. Bayesian optimization, on the other hand, provides a better trade-off between accuracy and scalability, making it a more suitable choice for large-scale predictive analytics in Spark MLlib.

5 Conclusion and Future Scope

This research compared traditional hyperparameter tuning methods (grid search, random search) with an advanced technique (Bayesian optimization) in Apache Spark MLlib. Using the NYC Taxi Trip Duration dataset, this paper evaluated the performance of these methods in terms of accuracy (RMSE) and scalability (execution time). The results show that Bayesian optimization achieves a better trade-off between accuracy and scalability compared to grid search and random search.The key contributions of this research are:A comprehensive comparison of hyperparameter tuning methods in Spark MLlib.Insights into optimizing hyperparameter tuning for large-scale predictive analytics.Future work could explore the implementation of other advanced techniques, such as Hyperband, and evaluate their performance on larger datasets. Additionally, optimizing resource utilization (e.g., CPU, memory) during hyperparameter tuning could further improve scalability.

Funding Declaration
No funding was received for conducting this study or preparing the manuscript.

References

1. Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A., Hyperband: A novel bandit-based approach to hyperparameter optimization, Journal of Machine Learning Research, (2018).

2. Feurer, M., Hutter, F., Hyperparameter Optimization, Automated Machine Learning, (2019).

3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., Optuna: A Next-generation Hyperparameter Optimization Framework, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (2019).

4. Real, E., Aggarwal, A., Huang, Y., Le, Q.V., Regularized evolution for image classifier architecture search, Proceedings of the AAAI Conference on Artificial Intelligence, (2019).

5. Klein, A., Falkner, S., Springenberg, J.T., Hutter, F., Efficient and Robust Automated Machine Learning, Advances in Neural Information Processing Systems, (2019).

6. Xie, Y., Zhou, Y., Scalable hyperparameter tuning in distributed environments, Journal of Machine Learning Research, (2020).

7. Shilpa P., Meenakshi G., Balancing between 'Big Data Analytics Tools', International Journal of Advanced Research in Computer Science, (2020).

8. Akshata B., Meenakshi G., Sentiment analysis on twitter data using lexicon-based and naive bayes approach, Journal of Data Mining and Knowledge Discovery, (2020).

9. Zheng, A., Casari, A., Feature Engineering for Machine Learning: Principles and Techniques, O'Reilly Media, (2018).

10. Raschka, S., Patterson, J., Nolet, C., Machine Learning in Python: Main Developments and Technology Trends, arXiv preprint arXiv:2002.04803, (2020).

11. Paleyes, A., Urma, R.G., Lawrence, N.D., Challenges in Deploying Machine Learning: A Survey of Case Studies, ACM Computing Surveys, (2022).

12. Bisong, E., Building Machine Learning and Deep Learning Models on Google Cloud Platform, Apress, (2019).

13. Géron, A., Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly Media, (2019).

14. Chollet, F., Deep Learning with Python, Manning Publications, (2018).

15. Karau, H., Warren, R., High-Performance Spark: Best Practices for Scaling and Optimizing Apache Spark, O'Reilly Media, (2017)

16. Kumar, A., Boehm, M., Yang, J., Data Management in Machine Learning Systems, Synthesis Lectures on Data Management, (2019).

17. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A., AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data, arXiv preprint arXiv:2003.06505, (2020).

18. Falkner, S., Klein, A., Hutter, F., BOHB: Robust and Efficient Hyperparameter Optimization at Scale, Proceedings of the 35th International Conference on Machine Learning, (2018).

19. Yang, C., Akimoto, Y., Kim, D.W., Udell, M., OBOE: Collaborative Filtering for AutoML Model Selection, Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (2019).

20. Wistuba, M., Schilling, N., Schmidt-Thieme, L., Scalable Gaussian Process-Based Transfer Surrogates for Hyperparameter Optimization, Machine Learning Journal, (2019).