Dataset Processing and Supervised Learning

Khai Truong
ktruong43@gatech.edu

***Abstract*—This paper explores supervised learning (SL) in machine learning (ML) by solving for two classification problems: one for credit approval decision, and one for car versus bus classification. Real world implications and prior knowledge about these datasets' subject matter were leveraged to inform our development process. In addition, their raw data format provides opportunities to experiment with feature engineering and selection techniques. We hypothesize that through straightforward fine-tuning for features and hyperparamaters, we can produce performance metrics of $> .9$ for accuracy, recall, specificity, and f1 score as our models are applied to the datasets.**

I. INTRODUCTION

This paper analyzes two datasets each with their own set of challenges when implementing supervised ML. The first dataset focuses on credit approval decisions, with 15 attributes, both nominal and numerical, with the target boolean attribute for whether credit was approved. The second dataset is a labeled GPS dataset, capturing latitude, longitude, and time attributes, with the target attribute being the instance was recorded on a car or a bus.

We are able to derive valuable insights and tradeoff decisions as we approach the datasets through the ETP framework: Experience, Task, Performance [1]. In the credit approval dataset, the significant financial loss of giving credit to eventual charge-off (creditees defaulting on their loan) introduces a design bias for high specificity to avoid deploying capital at a loss. In contrast, the GPS dataset aims to predict if the instance was on a car or a bus, with nothing but latitude, longitude, and timestamp as the base attributes. This presents a non-trivial challenge to derive more informative attributes from the raw data. These inherent biases and prior knowledge inform our model's Performance evaluation and thus kickstart our model development process.

In this paper, we implement six ML models: neural networks (NN,) support vector machines (SVM,) k-nearest neighbors(KNN,) decision trees (DT,) and random forest (RF.) Neural networks with various activation functions will be used to capture complex, non-linear relationships, making them well-suited for both the high-dimensional attributes of the credit dataset and the spatial-temporal data in the GPS dataset. SVMs, implemented with at least two different kernels (e.g., linear and radial basis function), will explore how different decision boundaries impact the classification tasks. KNN, applied with a large K value, will help smooth out noisy data, providing a comparison against more sophisticated algorithms when classifying car versus bus instances from GPS coordinates [1]. Additionally, AdaBoosting will be implemented for decision trees and we will also incorporate pruning techniques by setting max_depth to facilitate early pruning. Our hypothesis is simple: that through one round of feature and model tuning, we can achieve high classification performance (as defined by $> .9$ for accuracy, recall, specificity, and f1) on these credit approval and GPS trajectories labeled dataset.

II. DATA

This section describes available raw data and how they facilitate our supervised learning (SL) task. For reproducibility, both datasets (credit approval and GPS trajectories) can be accessed from UC Irvine's machine learning data repository.

TABLE I
DATASETS AND TASKS.

Dataset	Target Attribute	Attribute Categories
Credit Approval (18,107 rows)	Approve OR Disapprove	Numerical, Categorical
GPS Trajectories (690 rows)	Car OR Bus	Numerical

In the credit approval dataset, attribute names and values have been changed to meaningless symbols to protect confidentiality of the data. The dataset contains 15 attributes varying from continuous numerical to nominal categories. As a result, this dataset provides opportunities to experiment with feature encoding techniques with encoding and bracket the raw data to enhance model performance. The target attribute is of +,- for whether credit was approved, providing our labels to conduct SL. To simplify evaluation, we treat these labels as ideal and as such provide a true guide post for our models (e.g. having been hypothetically "validated" by consequential creditee payment history.)

The mix of numerical and nominal features require the validation of various feature encoding techniques like one-hot encoding, label encoding, and binning to transform the raw attributes. The labels +, - offer a classic binary classification task, and given that the dataset represents real-world decisions on credit, it can test how well models generalize to unseen data. Refining these data is part of a process called ETL (Extract, Transform, Load.) Our ETL's direction is determined by how our feature selection and engineering affect model outcomes.

The second dataset contains GPS pings from volunteer session where each recording has its own unique track_id and a constant mode of transport of either a bus or a car throughout their session. Each instances records latitude, longitude, and its timestamp. Location and time are insufficient to induce if a person is on a car or a bus. As a result, this dataset provides

opportunities for feature engineering to derive secondary attributes such as speed and acceleration by differentiating for rate of change as grouped by track_id [2]. The GPS dataset is compelling due to its demand to increase model complexity by deriving more attributes from the raw latitude and longitude.

III. METHODS AND MODELS

A. Feature Engineering

1) Feature Encoding: For the credit approval dataset, preprocessing begins with encoding nominal attributes using one-shot encoding to convert categorical variables into a binary format. Additionally, all attributes, both nominal and numerical, will undergo min-max scaling to normalize the values between a specified range. We expect scaling to also help avoid imposed ordinal relationships in the encoded nominal values.

2) Feature Engineering: Formally, feature engineering involves the application of transformation functions such as arithmetic and aggregate operators on given features to generate new ones [3]. However, this process can also be conducted manually using domain prior knowledge. For the GPS dataset, we derive two initial features: speed and acceleration, by calculating the rate of change in latitude and longitude across successive instances of the same track_id. A specialized function computes the speed based on the distance between two geographical points (using the haversine formula,) and the acceleration is simply rate of change of speed. This process provides time-series attributes to our dataset, providing our models a means to classify bus versus car.

B. Model Selection

This paper focuses on supervised ML which is to build an algorithm that can receive input data and use statistical analysis to predict outputs while being updated by adding new data [4]. We will implement six different SL models in this paper using our labeled data (see table II.) For reproducibility, we set random state parameters of any function calls in our code to the author's GT ID. The paper utilizes Python and libraries ideal for experimentation on small datasets such as Pandas, NumPy, and scikit-learn.

1) Performance Evaluation: We first establish a baseline by gathering performance metrics of a baseline set of models using default hyperparameters and features discussed above. The paper we primarily rely on the sklearn library to generate these metrics and visualized in the form of learning curve graphs to simultaneously check for overfitting. We utilize learning_curve from sklearn.model_selection and set cv to 5 to also implement 5-fold CV in our iterations.

In this paper's k-fold CV, the dataset is divided into **5 equal-sized folds**. The model is trained on k - 1 folds while one fold is held out as the validation set. This process is repeated k times, ensuring that each fold serves as a validation set exactly once. The sampling process also aim so that no two test sets overlap [5]. This method provides a comprehensive evaluation of the model's performance across different data distributions. This is aimed for model's robustness to prevent overfitting.

For the task of **credit approval**, we assert these following business requirements: that specificity and recall are the two main metrics to maximize. This is because (1 - specificity or 1 - $\frac{TrueNegatives}{TrueNegatives+FalsePositives}$) is directly the charge-off rate leading to avoidable financial loss, while (1 - recall or 1 - $\frac{TruePositives}{TruePositives+FalseNegatives}$) represents clear opportunity costs of an avoidable missed customer.

2) Model Tuning: All analyses and graphs generated during feature engineering and model tuning are from **only the training set**. This is to prevent peeking where we are introducing design decisions that are also influenced by "unseen" test data and thus put the models' generalization capability at doubt. With this operating principle, we then generate performance metrics for our models at their default hyperparameter values. Once a baseline performance is obtained, we then utilize RandomSearchCV from sklearn.model_selection to perform a preliminary search for hyperparameter values across a set of designated performance metrics for classification models: "accuracy", "precision", "recall", "f1", "roc_auc", and "specificity".Our target metrics to optimize for are listed in table II.

Possible tuning steps will be to incorporate regularization techniques to address potential overfitting and improve generalization. Regularization adds a penalty to complex models, encouraging simpler models that generalize better to unseen data. This paper will experiment with applying L2 regularization (Ridge) to models such as Support Vector Machines, so as to tune the regularization parameter (e.g., C for SVM and alpha for Ridge) and aim to balance model complexity with predictive performance. By adjusting these parameters, we aim to improve our models' ability to generalize beyond the training set, especially for datasets with a high number of features or potential noise. These steps are recapped in the model development flow chart below.

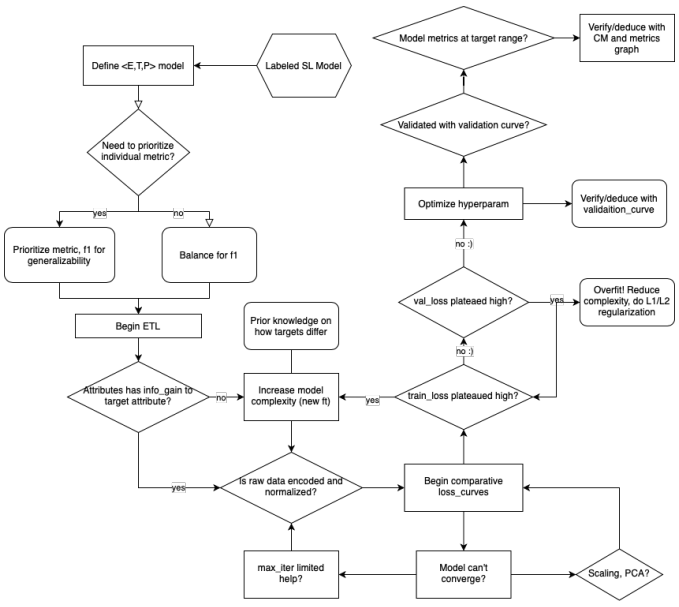

Fig. 1. Model Development Flow chart

Finally, we will use validation curves to visualize and validate the impact of regularization and hyperparameters tuning. Validation curves provide insight into how the model's training and validation scores change as the hyperparameters are adjusted. We will test for key metrics such as n_neighbors for our KNN model (e.g. a smaller n focuses on closer, potentially noisy neighbors leading to overfitting, while a larger n smooths predictions by considering more distant neighbors, which can result in underfitting) and C values for SVM (e.g. a small C penalizes misclassifications, resulting in a simpler decision boundary, while a large C allows for more complex decision boundaries, potentially leading to overfitting.) By plotting training and validation scores against different hyperparameter values, we can identify the sweet spot where the model is neither underfitting nor overfitting. The intended outcome of this stage is a learned table of "optimized" values of our model hyperparamaters that yields the most robust and balanced performance across various metrics such as accuracy and f1.

IV. RESULTS: CREDIT APPROVAL

A. Credit Approval — Baseline

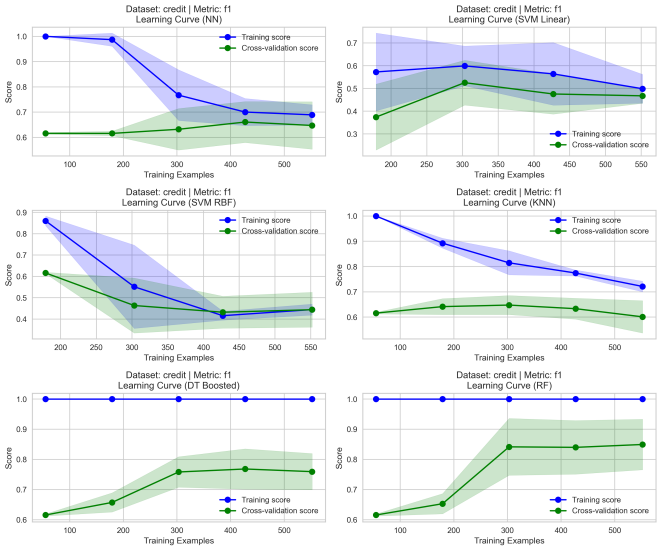

Fig. 2. Baseline metrics — Credit Approval

A preliminary observation is SVM linear failing to converge in a reasonable timeframe. As a result, we implement max_iter = 1000 as a short-term measure and plan to explore scaling in later iterations to resolve. To troubleshoot for this, we aim to implement **scaling** as an additional step in our feature engineering. Scaling (normalization) is crucial especially with linear kernels, because SVM optimization relies on calculating distances between data points. If features are on different scales, the model may struggle to converge, leading to suboptimal performance or failure to find a solution within a reasonable timeframe. By normalizing the data, we ensure that each feature contributes equally to the distance calculations, preventing any single feature from disproportionately influencing the model. To benchmark, the average runtime for generating learning curve plot was 50 seconds when SVM linear has max_iter = 1000. The average runtime after scaling was implemented and max_iter removed was 67 seconds. This increase of roughly 33% in runtime suggests a trade-off between computational efficiency and model performance. Initially, the SVM linear model struggled to converge within 1000 iterations (model took 72 minutes to finish.) By removing the max_iter limitation, we allowed the model to converge without a predefined cutoff, which is beneficial for performance but also demands more computational resources as it explores a broader parameter space. However, the trade-off led to improved model performance and stability. As a result, we decide to keep scaling implemented in our models.

We also note that training and CV curves failed to converge properly, indicating possible variance impacting our models' performance. As our models tend to overfit (100% in training for some models,) this represents an opportunity to improve performance by tuning hyperparamaters and introduce regularization.

B. Credit Approval — Tuning

We now use RandomizedSearchCV to derive hyperparameter values as scored by f1, accuracy, and specificity. We choose this method over GridSearchCV as by randomly sampling a predefined number of parameter combinations, we can identify the most promising configurations within a limited timeframe. This approach accelerates the search process and reduces the risk of overfitting, as it evaluates the model's performance across various parameter settings and validation folds. The results of this optimization process are illustrated in the accompanying graph, which demonstrates the improved performance metrics of the models at modified hyperparameter values.

Obtained results demonstrated improvement. However we notice high variance in our performance metrics. A large standard deviation around the learning curve suggests that the model's performance was highly dependent on the specific data samples used for training and validation. This inconsistency can make it difficult to accurately assess the model's true capabilities and generalization potential. As a result, we next aim to control for this by implementing **stratified CV** as we generate our learning graph and models. Stratified CV ensures that each fold maintains a similar distribution of target classes as the overall dataset. This technique is particularly useful when dealing with imbalanced datasets or when certain subgroups within the data are critical for model performance. Indeed, mitigating for false positives is important for our credit approval task as discussed. Once implemented, we successfully obtain a tightening of the standard deviation area around the learning curve as seen in Fig. 4 and Fig. 5. As a result, we keep this implementation of stratified CV in subsequent models.

C. Credit Approval — Validation Curve

Now that we have obtained improved model performance, we seek to validate this gain by inspecting our hyperparameter

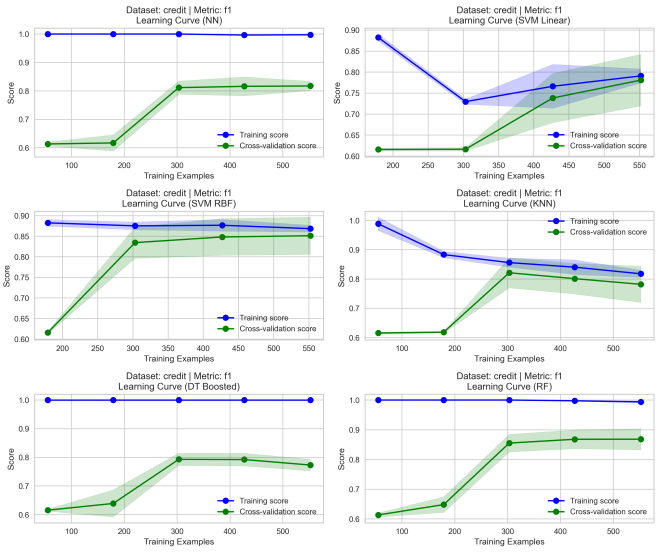

Fig. 3. Optimized & Stratified metrics — Credit Approval f1 score

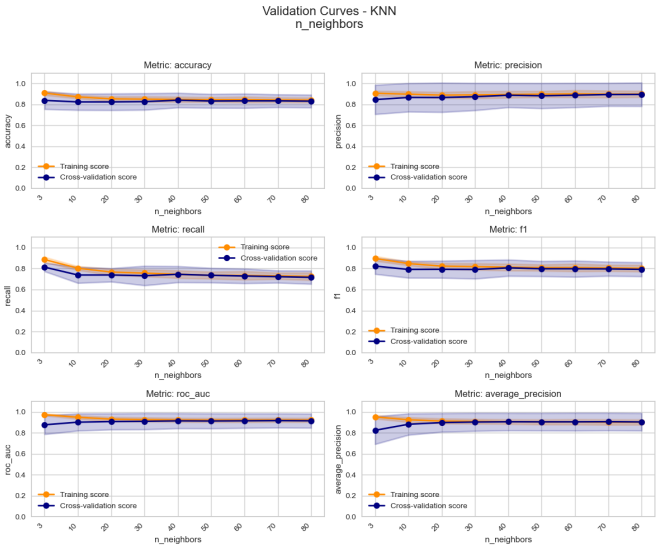

Fig. 4. n_neighbors validation curve by metric — Credit Approval

selection in validation curve graphs. For the credit approval dataset, the following values were utilized. To set an example, we explore in detail the validation process with our KNN model as varied by param_name = n_neighbor. Validation graphs demonstrated that n_neighbors = 40 for the KNN model in the context of credit approval strikes an optimal balance between overfitting and underfitting across various performance metrics. Results show that training and cross-validation scores for accuracy, precision, recall, and F1 score stabilize at 40, indicating good generalization. As a result, we expect at this value the model captures most creditworthy applicants without introducing too many false positives, which is crucial for maintaining financial viability. As a result, we gain confidence to keep 40 as our hyperparameter value in subsequent iterations.

We repeat this manual validation process with other models' values for the dataset. A case study of note is the values for the hyperparamater C - a regularization parameter for our SVM models which demonstrate an optimal value at 0.1. This suggests that a moderate level of regularization is effective in balancing the trade-off between bias and variance, preventing the model from overfitting while still capturing the essential patterns in the data. The choice of C influences how the SVM handles misclassified instances: if C is too small then insufficient stress will be placed on fitting the training data, if C is too large then the algorithm will overfit the training data [6].

D. Credit Approval — Classification Performance

Now that we have considered and modified our features and models, we move onto reviewing if we have obtained our hypothesized model satisfying target of ¿.9 for accuracy and specificity and ¿.8 for f1 score. In addition, we remind ourselves of the credit approval task bias of avoiding charge-

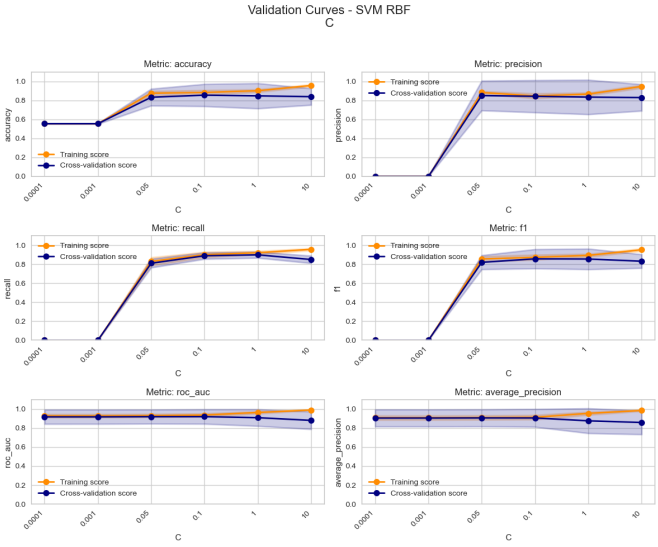

Fig. 5. C value validation curve — Credit Approval

off $falsepositives$ and capturing as many clients as possible $falsenegatives$. Specifically,

While K-fold CV (kFCV) is robust, we only implement k = 5 so that the resulting box plots only represents 5 data points. To complement kFCV, Monte Carlo cross-validation (MCCV) offers an alternative approach [7]. In MCCV, the dataset is randomly split into training and test sets multiple times—designated 100 iterations in our paper—where a specified fraction of the data, such as 80%, is used for fitting. Each iteration creates a different random split, allowing the model to be evaluated across various subsets of the data. The resulting MCCV plots show results that align with kFCV metrics, indicating these results to be reliable.

In the k-fold CV results, our models demonstrated favorable

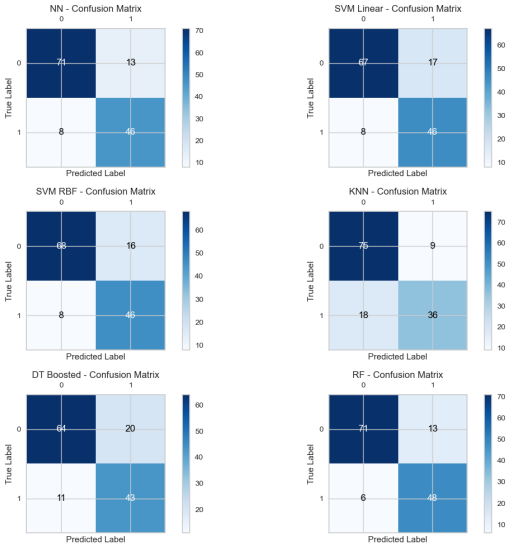

Fig. 6. Confusion Matrix all models — Credit Approval

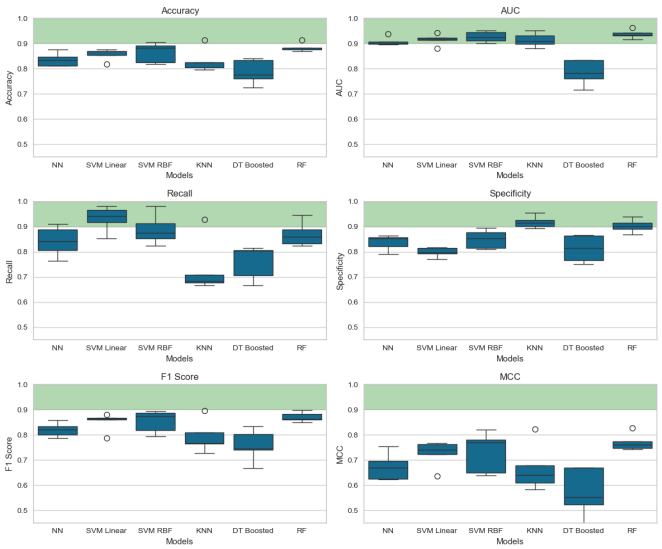

Fig. 7. kFCV Models Performance — Credit Approval

results in our graphs where the area [.9, 1.0] is colored green to signify target zone. The models' strong AUC indicates positive overall performance and good discrimination between the positive and negative classes. The Matthews correlation coefficient (MCC) of 0.70, while indicative of a reasonably good model, suggests that there may be some imbalance or errors in classifying the minority class. To standardize model selection, we first look for models with best f1-scores which indicate an superior balance between precision and recall. As specificity represents direct financial loss, we thus disqualify models with good f1 but underperforming specificity such as SVM linear. This leaves us with SVM RBF and RF as our two best contending models. We judge these results to be reliable and generalizable as kFCV and MCCV results in the training set align with metrics obtained when models predict for only the test set data. Unfortunately, this does not confirm our thesis that a high classification performance can be achieved in one round. To further enhance model performance, the next round of feature and model tuning could include revisiting available attributes and conduct further feature engineering. Indeed, this is something we will focus on for our second dataset with GPS trajectories.

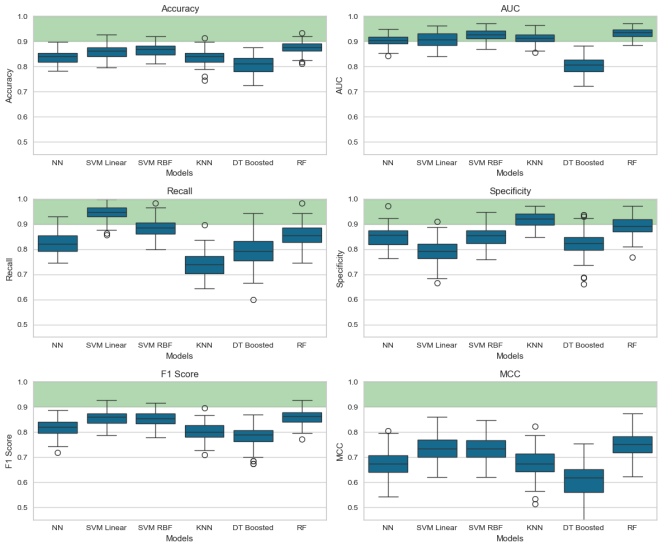

Fig. 8. MCCV Models Performance — Credit Approval

V. RESULTS: GPS TRAJECTORIES

A. *GPS Trajectories— Baseline*

While the first dataset has the benefit of implicit feature engineering already been done by an established finance industry with sophisticated attributes to predict credit worthiness, our second dataset is comparatively primitive. Given latitude, longitude, timestamp, track id, and labeled target of car or bus, we are to derive additional features to predict such labels. Additional information is needed to make a viable induction. We tentatively create two additional features through calculating rate-of-change: speed and acceleration.

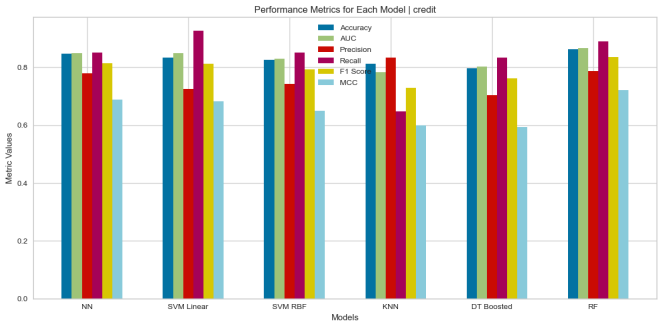

Fig. 9. Test Set only Models Performance — Credit Approval

Initial model performance underperformed significantly with these two derived features.

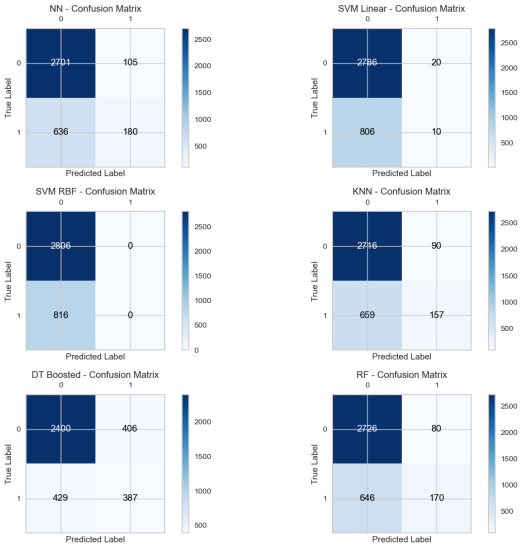

Fig. 10. Models Performance with 2 Features — GPS Trajectories

B. GPS Trajectories — Feature Engineering

Additional features are needed to support an inductive basis for the model to reasonably classify bus or car. With the raw data having only latitude, longitude, and timestamp, we are left with limited options. Speed and acceleration also do not yield further predictive power as both cars and buses can move at similar speed. We attempt to build an **inductive model** of how a person would reasonably differentiate travelling on a car versus a bus. Based on the prior knowledge that buses tend to only operate during business hours, we induce that knowing when during the day the GPS location was recorded would be informative [2]. As a result, we attempt to derive another feature called hour_of_day from the raw timestamp value. Initial runs demonstrate clear improvement on performance.

Based on this improvement, we continue to rely on our inductive model to inform further informative features. What other factors differentiate being on a bus versus a car? Indeed, if hour of day provides predictive power, then we can also expect day_of_week to also be of help as bus trips are less common on the weekends. In addition, a bus would make more frequent and longer stops to pick up passengers. These attributes can be derived from the speed attribute, keeping a check on how many stops there have been based on each recording where speed value drops near 0. As a result, we are able to introduce a running_sum value of number of stops and running_avg value of stop length. We do not calculate total sum and total average per track_id to **avoid peeking** as that could introduce information on ”future stops” into the current instance row. Obtained results in Fig. 13 are satisfactory, with DT and RF models nearly performing perfectly.

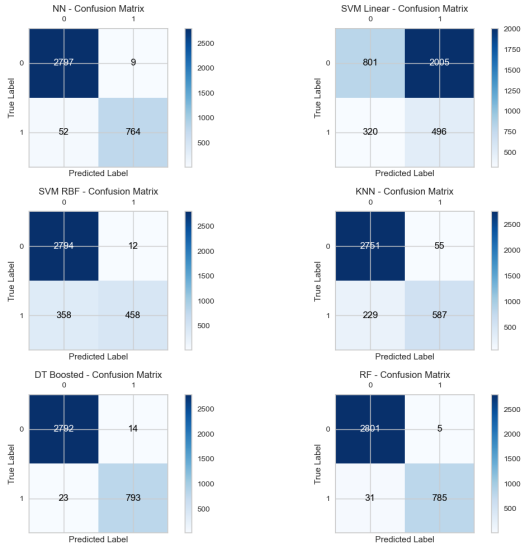

Fig. 11. Models Performance with 6 Features — GPS Trajectories

C. GPS Trajectories — Tuning

Now that we have leveraged our inductive model to derive informative attributes, we apply tuning techniques utilized in the credit approval dataset. We first start with implementing scaling to facilitate model convergence particularly with SVM linear kernel and NN. Subsequent graphs utilize scaled features. Similar to the previous dataset, we also implement RandomizedSearchCV to obtain a starting set of ”optimized” hyperparameters. Indeed, the observation that models such as DT and RF can achieve near perfect results while SVM and KNN models underperform suggest that our models are simply incorrect for a tractable classification problem. These possible misconfigurations makes results obtained from RandomizedSearchCV promising.

While we can graphically analyze these obtained values using validation curves, here we briefly note that tanh was chosen over relu for NN’s activation function for the GPS dataset. This is likely because our scaled continuous numerical features. These features could have negative values (acceleration, speed, etc.), and tanh would be better at handle the negative values and their nonlinearities compared to relu, which zeros out all negative inputs. As a result, we keep tanh as our activation function.

In terms of performance to sample size with accuracy as a benchmark metric, the high value of training performance suggests possible high variance. However, we observe convergence for training and CV curves which indicates good overall model performance and a balance between bias and variance.

D. GPS Trajectories — Validation Curve

In this section we manually cross-check the values obtained from RandomizedSearchCV and their corresponding validation curve. Mainly, the variation of KNN’s n_neighbor, SVM’s C values are interesting. For the KNN hyperparamater, we note that n_neighbors = 10, the model obtains better accuracy,

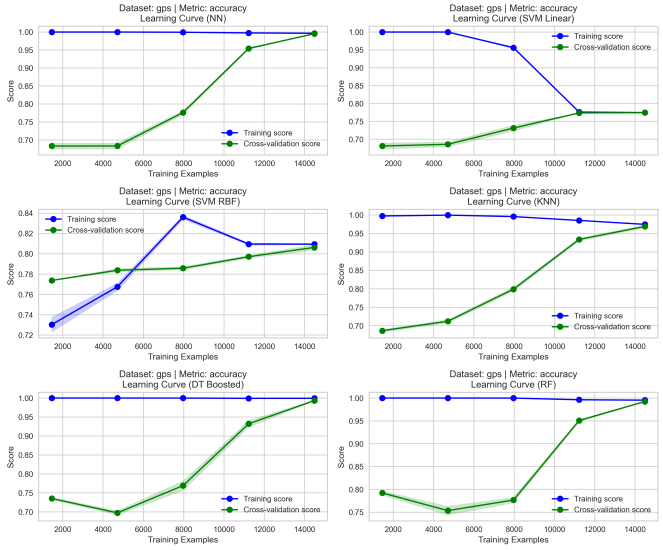

Fig. 12. Learning curve for accuracy — GPS Trajectories

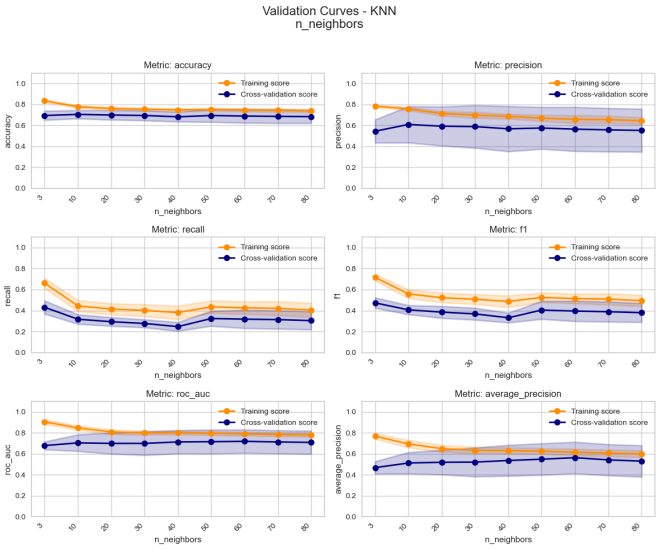

Fig. 13. n_neighbor validation curve — GPS Trajectories

precision, and roc_auc. For n_neighbors = 3 the models achieve better recall and f1 score. We would not prioritize recall here as this comes at a cost of more false positives. As there are no immediate penalty to misclassify between car or bus, we opt for the increased generalization capability that roc_auc and accuracy brings with n_neighbor = 10. For the SVM models, the C parameter controls the trade-off between a smooth decision boundary (margin) and correctly classifying training points [6]. A lower C (e.g., 0.01) emphasizes a wider margin but allows for more misclassification, while a higher C (e.g., 10) focuses on minimizing classification errors but may lead to a narrower margin and potentially overfitting. Our validation curves shows at C = 0.1 the best rocauc and accuracy are obtained. This implies the models are likely generalizing well, handling both the noisy and well-separated data points in the training set effectively.

E. GPS Trajectories — Classification Performance

NN, KNN, DT, and RF all performed adequately with high results. These models confirm our hypothesis that a high performance can be achieved through tuning in one round. Indeed, the tight box plot distribution for kFCV of models' performance suggest very consistent performance with great balance for bias and variance.

Of note is the severe underperformance of SVM models for this dataset. This suggests that bias (particularly bias in the attributes,) not variance, is driving SVM's underperformance. We suspect that either that both linear and rbf kernels are a bad fit, or that the feature scaling resulting in negative values were the root cause. A sample run of a poly kernel also yielded similarly low results.

VI. DISCUSSION

There were techniques and possible checkpoints we did not touch upon, such as applying Principal Component Analysis

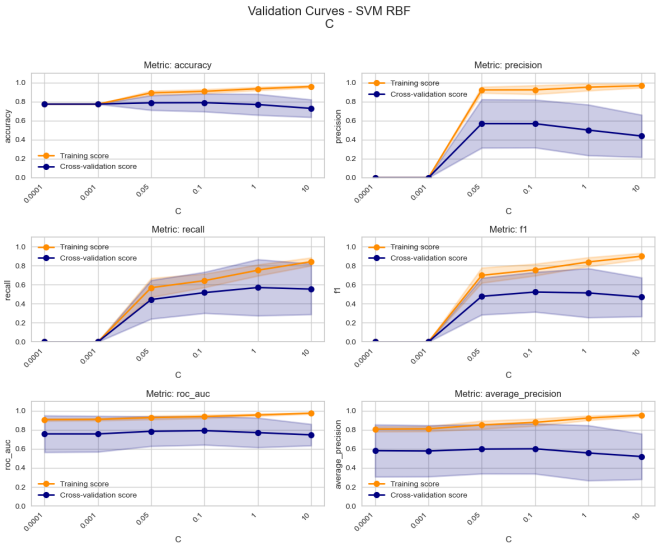

Fig. 14. C value validation curve — GPS Trajectories

(PCA) as part of our ETL process. This is because a model that uses data filtered through PCA would most likely require its own round of tuning which makes it inapplicable to our first round of exploring the credit approval. Meanwhile, the GPS dataset involves adding derived features, which makes PCA irrelevant for the first round of development. Nevertheless, it can be used in subsequent rounds to eliminate which derived features were less useful. Future iterations of the GPS dataset models can also implement feature importance to verify and understand which derived features are contributing.

We also did not review loss curve graphs for our neural net model, as interestingly NN did not show distinct performance in these two datasets. This underperformance was likely driven by the credit approval dataset limited size of 688 instances,

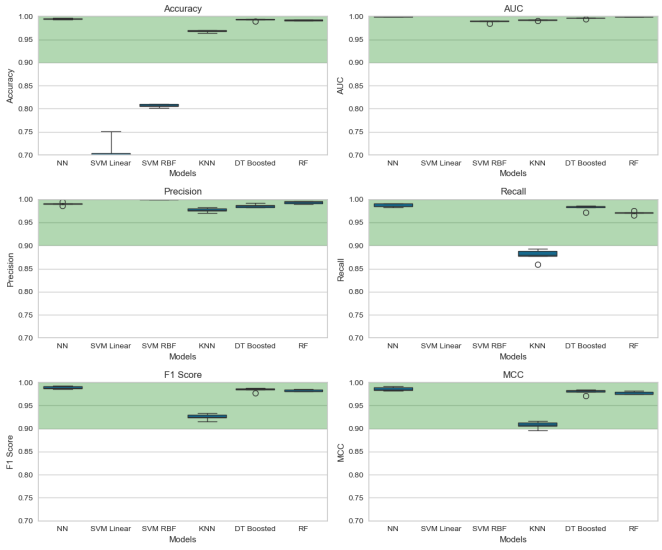

Fig. 15. kFCV Models Performance — GPS Trajectories

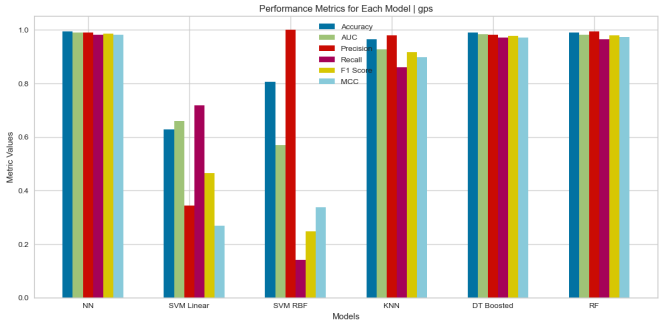

Fig. 16. Test Set only Models Performance — GPS Trajectories

while the GPS dataset was poor in terms of complexity with only 6 attributes. Indeed, simpler models such as SVM and RF easily performed, demonstrating their capability to generalize in more limited datasets.

The discovery of SVM underperformance for the GPS dataset also invoked interesting lines: choice of kernel, feature scaling, and the inherent complexity of the data. As we have tested briefly for the poly kernel as well as plotting out validation curves for C and gamma values, we narrow it down to possibly feature scaling as the root cause. Furthermore, analyzing the feature importances and exploring additional features may enhance SVM's capability in subsesquent rounds of tuning. However, this direction serves little practical use as we have already obtained other models that perform well for the dataset.

VII. CONCLUSION

In conclusion, our hypothesis that high classification can be achieved within one round of feature and model tuning was partially confirmed. In the credit approval dataset, we were impacted by high variance where models overfit and learning curves failed to converge. In the GPS dataset, our SVM models were possible impacted by high bias in the attributes. In both datasets, we needed to leverage prior domain knowledge to identify performance metrics and derive features. We were able to confirm our hypothesis by adhering to methodological steps to define ETP, implement ETL, review baseline, refine hyperparamaters, validate though CV and test set, and identify opportunities from bias or variance to troubleshoot in subsequent rounds as the entire sequence of steps is repeated. Ultimately, this paper underscores the importance of domain knowledge and an iterative approach as we tackle future machine learning problems.

VIII. REFERENCES

REFERENCES

[1] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.
[2] H. Orami, "Predicting travel mode of individuals by machine learning," in *Transportation Research Procedia*, vol. 10, (Delft, The Netherlands), pp. 840–849, Elsevier, 2015.
[3] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. Turaga, "Learning feature engineering for classification," *Computer Science*, 2023.
[4] M. Zafari, A. Sadeghi, S.-M. Choi, and A. Esmaeily, "A practical model for the evaluation of high school student performance based on machine learning," *Applied Sciences*, vol. 11, p. 11534, December 2021.
[5] D. Berrar, "Cross-validation," in *Reference Module in Life Sciences*, Elsevier, 2018.
[6] R. Burbidge and B. Buxton, "An introduction to support vector machines for data mining," *Computer Science*, 2001.
[7] Q.-S. Xu and Y.-Z. Liang, "Monte carlo cross validation," *Chemometrics and Intelligent Laboratory Systems*, 2000. Received 6 July 2000; accepted 27 November 2000.