Optimal Model Selection for Signal Pattern Recognition and Production Workflow: A Hill vs Valley Case Study

Carlos Juarez

Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA

Abstract

The ability to quickly classify a graph based on its curvature is a key motivator for many companies to utilize machine learning models in their data post-processing. For this task, a noisy dataset, consisting of "hills" (1) and "valleys" (0), was transformed through smoothing and scaling techniques in preparation for machine learning models. The dataset went through multiple ML model fits (Logistic Regression, Support Vector Machines, Decision Trees, Random Forest, XGBoosting) and went through cross validation techniques to find the best hyperparameters to output their best evaluation metrics. A decision for the best model was made based on a higher recall value due to the company's need to output more true good devices. While Logistic Regression with smoothing and standard scaling output the highest test accuracy, precision score, and specificity score, the XGBoost model was selected as the superior model due to the need to have a higher recall score to meet production needs.

Introduction

Classification algorithms are an important key in a company's production timeline for determining if a device is classified as "pass" or "fail" and ready for further testing. In this algorithm problem, curves will be classified as a "hill" (1) or "valley" (0). Although at face value, this is presented as a simple classification model, classifying a device as pass or fail based on the curve of a graph determines if a device can proceed to further testing. For example, a capacitance vs voltage (CV) curve is considered a "pass" if the capacitance vs voltage curve follows an exponential trend (1) or "fail" (0), otherwise. This task is necessary for many engineering principles, specifically for quality control, hitting deliverables, and optimizing product output. In a professional environment, the amount of data collected exceeds the possibility of an engineer to correctly classify a graph trend. This is why it is important to develop an ML model that accurately classifies curves to decrease human error and increase production time.

A major challenge with the data is its high dimensionality. Because the datasets are curved, simple linear models such as linear regression or logistic regression are far less complex and cannot capture the curved shape effectively. The presence of noise in the dataset is also expected. Data collected in microfabrication labs tends to be noisy, coming from the semiconductor analyzer through thermal noise, flicker noise from low frequency measurements, or electromagnetic interference from wire connections. It can even come from walking around the system and causing any setup, such as pins in a contact pad, to vibrate. In the real world, we will need to find a way to transform the data to mitigate the noise. The last obstacle is creating a model that is generalized beyond the "hills" and "valleys" data. The actual data that will be used for this model is exponential, so the final code must be able to input exponential data without error.

Background

Devices produced from a wafer go through characterization techniques to determine if the device is good for mass production or packaging. Some of the main characterization tests include C&I, where a voltage sweep is applied through two connecting nodes and the output voltage is measured. Through Ohm's law, the resistance is calculated with the goal of seeing a high resistance/open circuit for nodes that should never connect or a low resistance for nodes that are expected to be connected. Another characterization test is the Capacitance vs Voltage (CV) test. This test will be the focus of our problem formulation and the algorithm. In a CV test, the goal is to apply a voltage sweep across the Proof Mass nodes to each appropriate nodal axis. As the voltage increases, the capacitance exponentially increases. In faulty devices, the capacitance flatlines or "gets stuck" as the voltage increases. Each wafer has around 350 devices, and each device has 9-12 CV curves to go through and classify. That is a lot of data to go through and classify. With our devices heading towards production, there is a need for a post-processing script that can accurately classify devices based on the CV curve. It is also important to readily output the data so that the devices that "pass" can go through other tests without the need to test on "failed" devices. Devices that fail this characterization can then go through R&D to identify any metrics of faults. To follow classified information restrictions, a similar dataset from the UC Irvine Machine Learning repository called “Hill-Valley” will be used (Graham and Oppacher 2008). The concept will be the same; the models will analyze the data and predict if the curve is a “hill” (1) or a “valley” (0) or, in CV test terms, if the curve exponentially grows (1) or doesn’t (0).

Learning Algorithms: Baseline Models

For this classification task, the baseline models will include Logistic Regression, Support Vector Machine (SVM), and Decision Tree.

Logistic Regression is one of the foundational classification algorithms, where it computes a linear combination of the input features called the score or logit (z). This score is then passed through a sigmoid function that maps the values between 0 and 1. The probability is the interpretation of the sigmoid function, specifically the probability that an output regression is the positive class or 1 (Lee, Fangfang. n.d.). A threshold is set to determine the final class prediction. Logistic Regression is fast to train due to its simplicity, provides a strong baseline for other classification techniques, and can relatively handle high-dimensional data, especially with ridge or lasso regularization.

Support Vector Machine or SVM is a supervised ML algorithm used for classification where the model is tasked to find the optimal boundary/hyperplane that separates the data points into different classes (Kavlakoglu, Eda n.d.). Although this model is used for both linear and nonlinear dimensions, non-linearly separable data can cause concern due to overfitting. Kernel functions are used to enable linear separation and decrease the complexity of the data, reducing overfitting.

A Decision Tree classifier is a supervised learning method with the goal of creating “if or else” statements based on the dataset (Decision Trees. Scikit Learn n.d.). The decision tree is easily visualized and simple to understand. The tree is grown in a top-down, recursive fashion. The root node of the tree is identified through information gain, evaluating every available feature to see which child nodes produce the purest split. In other words, information gain measures how much uncertainty or entropy is reduced after a split has been made. There is little need for preprocessing in this model, but for consistency, the Column Transformation was implemented. A Decision Tree can handle non-linear patterns that are common in the dataset through recursive splitting. The biggest disadvantage, even for any dataset outside of this example, is that the tree can easily become too complex due to the noisy dataset. There is a slight bias towards the majority class, however, the data itself has a split even number of hills and valleys.

Learning Algorithms: Ensemble Methods

For this classification task, the ensemble methods will include Random Forest and XGBoost

From decision trees, there is an ensemble machine learning algorithm that combines the output of multiple decision trees to reach a single result. This model is called Random Forest (Kavlakoglu, Eda. n.d.). Specifically, this estimator uses bagging, fitting several decision tree classifiers over a sub-sample of the dataset and use averaging to improve accuracy and mitigate overfitting, a consistent issue with decision tree model. There is no feature scaling needed, like a decision tree, which reduces computational time, and can handle missing values through potential gain calculations. This model also helps with noisy data sets due to a majority vote rule across multiple trees (RandomForestClassifier. Scikit Learn. n.d.). Noisy data points that affect a few trees will not affect the final prediction.

Still in the same realm as decision trees, XGBoost is another model that builds its predictions by combining sequences of weak decision trees called base learners. It provides parallel tree boosting, where a single weak model is lifted or supported by several other weak models (XGBoost. NVIDIA. n.d.). For example, after a first decision tree is made, a second tree is built to correct any errors that the first tree has missed. After the second tree is made, a third tree is built to correct the errors from the second tree, and so on and so forth. This provides a sequence that focuses on data points that were not flagged appropriately. Weak trees are often “decision stumps” or decision trees with only one split. The loss function is used to evaluate which decision stump is the weakest and needs help. The goal is to have a significant, fast decrease in the loss function as the sequence proceeds. This is like a gradient descent, specifically Newton's method, where the current (n) ensemble is equal to the prior (n-1) ensemble plus the learning rate times the weak learner at (n-1) ensemble. XGBoosting for this dataset will be used because it bypasses any smoothing technique or feature scaling, which has been causing a problem in computation time. The noisy data will be handled from built-in regularization techniques, preventing a tree from being too complex.

Related Work

In a paper published by the MATEC Web of Conference called “A Comparative Study of Feature Selection Techniques for Bat Algorithms in Various Applications” by Rozlini, Mohamed, Munriah Mohd Yusof, and Noorhaniza Wahidi, feature selection was emphasized for its problematic approach. Feature selection uses a subset of features rather than all the features from a dataset to increase the evaluation metrics of a model. In their study, the aim is to use a feature selection algorithm known as the Bat

Algorithm to determine which features are crucial and if there is a significant change in the evaluation metric post-Bat Algorithm. The Hill and Valley dataset was used for the study, and the results were that using a feature selection algorithm showed little to no difference in their evaluation metrics from the baseline (Mohamed, Yusof, and Wahidi, 2017). Based on this note, the dataset for our study is used completely and has no feature selection done for any models.

Problem Formulation and Dataset

The first step of this dataset is to look at the raw data as a whole and find initial parameters. The data set is split into 4 categories: training data with noise, without noise, and testing data with noise and without noise. For this ML model, the focus will be on the noisy data set, as CV data tends to be noisy and needs noise reduction techniques. In the excel spreadsheet, there are data points from X1:X100 with the target column after X100. This means there are 101 features; 100 represent the magnitude of the Y at X coordinate, and the last represents the target label. The spread of the Y magnitudes ranges from 0 to 156612, and each set of curves has a varied range. This signifies that we will need to scale the data appropriately. For this dataset, there is no missing data. If there were missing data, the appropriate approach is to use Next Observation Carries Backward (NOCB), which is an imputing technique that replaces the missing value with the next available value. Using other methods, such as Mean/Median Impute, especially where the magnitude of the y value at position x is an important feature, i.e., hill or valley, would lead to misclassifications. In total, there are 1212 instances with 100 features each. Due to the size of the data, k-fold cross-validation will be used to prevent the model from overfitting.

For the noisy dataset, filtering techniques were applied to smooth out the transposed data. One of the techniques is called Simple Moving Average (SMA). A simple moving average is a technical analysis tool that calculates the average of a dataset over a period. This will "smooth" out the graph by using the averages within a set period and setting the average as the new output. It is important to provide an appropriate window size for the averages to smooth the graph while still maintaining its key features. For the baseline models, a window size of 5 will be used due to the number of columns pre-transposed for each feature (100). Cross validation to find the best window size for each model will be implemented later.

Another filtering technique is called LOWESS Smoothing (Locally Weighted Scatterplot Smoothing). This technique is good for non-linear trends. It has a parameter called frac (0,1) that controls the degree of smoothing by multiplying the frac by each point and estimating the new y using a weighted linear regression. The frac parameter used for the baseline models will be 0.5 because, although there are rapid local changes in the dataset, increasing the frac parameter can lose the main features that will be used for classification. In Figure 1-4, comparing the two methods, SMA smoothing keeps more of the distinct classification features, while LOWESS smoothing significantly decreases the features. SMA keeps the integrity of important features, keeping key turning points, and subtle features that complex models can "find" to distinguish classification. Although LOWESS smoothing significantly reduces the feature, this can provide simpler models with the advantage, allowing an increase in productivity output. The global trend is much easier to see with LOWESS smoothing allowing faster effective optimization techniques such as gradient descent.

ML algorithms such as Linear Regression or SVMs are sensitive to the scale of the numerical features. For our dataset, 2 feature scaling techniques will be applied: Standardization and Min-Max Scaling. Standardization Scaling scales the dataset to zero mean and unit variance. Min-Max Scaling scales the data set to a [min, max] range. The min-max scale will be set to a min of 0 and a max of 1. Scaling the graph appropriately is crucial for increasing convergence time and making the features more interpretable. There are four column transformers created based on data smoothing and scaling: SMA smoothing + Standardized scaling, SMA smoothing + MinMax scaling, LOWESS smoothing + Standardized scaling, and LOWESS smoothing + MinMax Scaling.

Methodology

The skeleton of each model's pipeline consists of a preprocessor and the model itself. This preprocessor is known as the ColumnTransfer and consists of the 4 column transformers as previously mentioned. Each model has its own pipeline. The dataset will go through a column transformation and its respected model (Figure 5). They will later be tuned for their best hyperparameters. The following models will be used for:

- Logistic Regression (LR)
- Support Vector Machines (SVM)
- Decision Tree (DT)
- Random Forest (RF)
- XGBoosting (XGB)

In Logistic Regression, the hyperparameters that will be tuned are the "penalty", which includes the regularization term to the cost function, the "solver", which finds the optimal weights through convergence to minimize the loss function, and "c", which will help tune the overfitting of the model. Since the data is high-dimensional and prone to irrelevant

Figure 1: Noise Reduction using Simple Moving Average on Hill (1) Classification

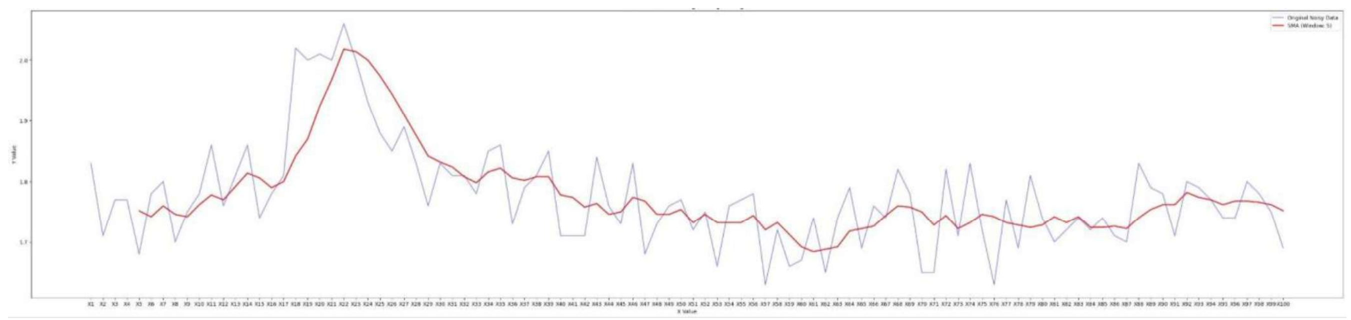


Figure 2: Noise Reduction using LOWESS on Hill (1) Classification

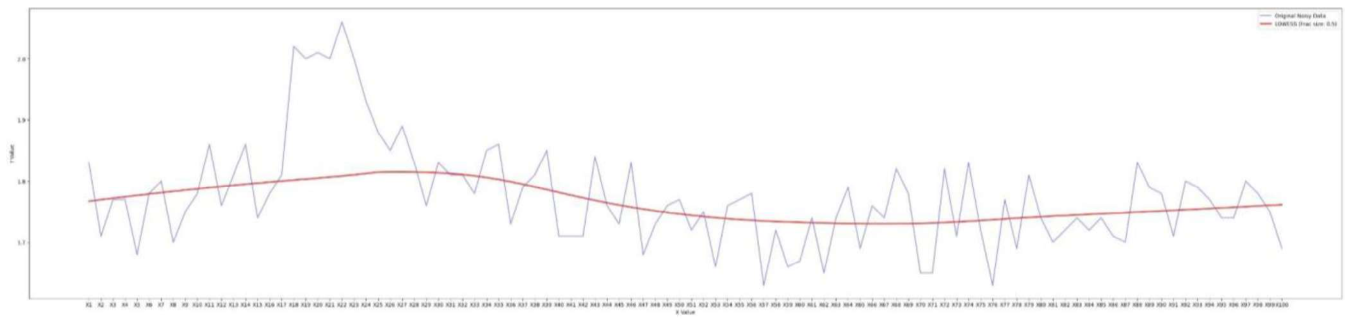


Figure 3: Noise Reduction using Simple Moving Average on Valley (0) Classification

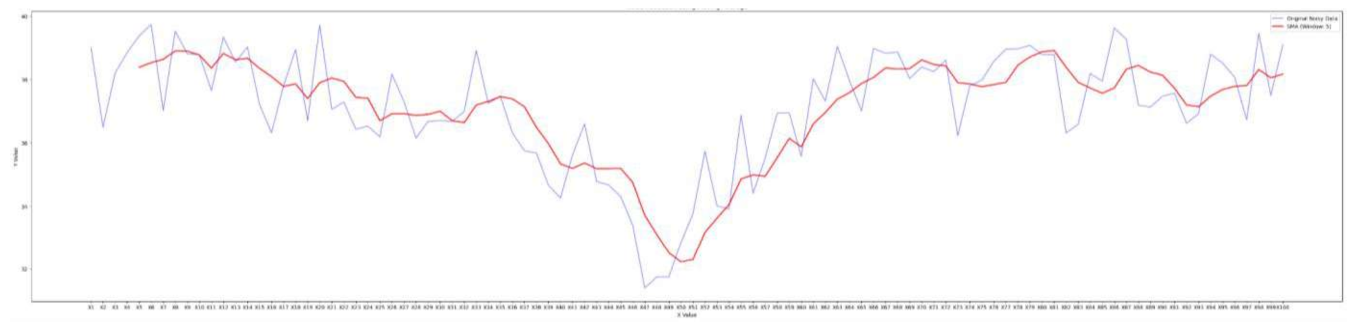
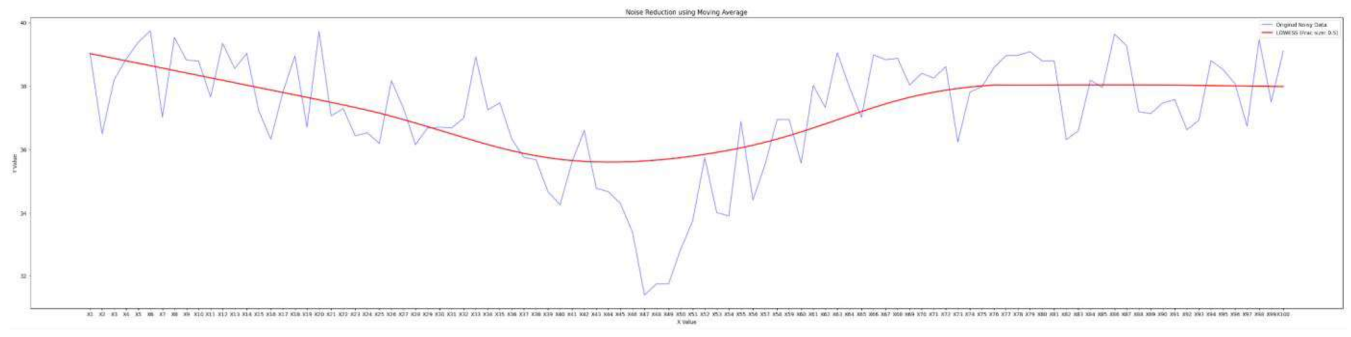
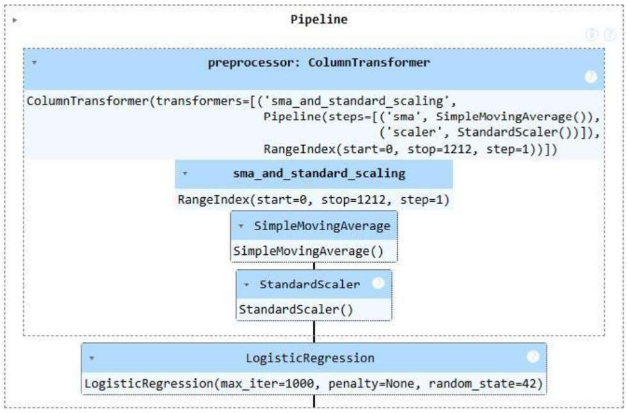


Figure 4: Noise Reduction using LOWESS on Valley (0) Classification



Features, it is important to tune the "penalty" hyperparameter. The "solver" hyperparameter such as Stochastic Average Gradient or L-BFGs can reduce training time significantly. Since the data set is relatively small, using L-BFGs over SAG is recommended especially if the "penalty" hyperparameters are being used. SAG is limited to L2 regularization because it requires the function to be differentiable at zero. The "max iteration" will be set to 1000 due to preventing reaching iteration limits with the "solver".



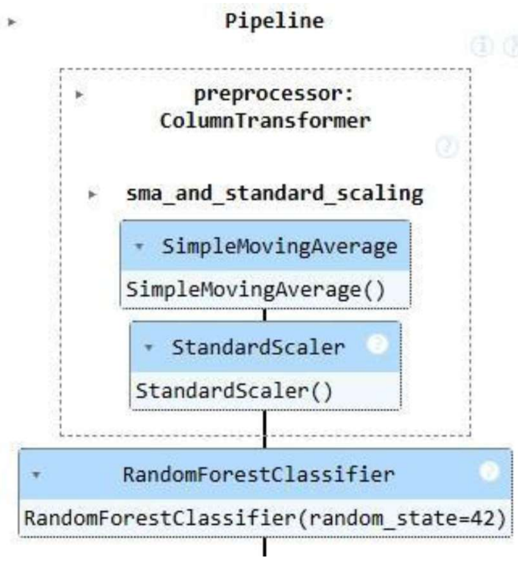
```
ColumnTransformer(transformers=[('sma_and_standard_scaling',
                                 Pipeline(steps=[('sma', SimpleMovingAverage()),
                                                 ('scaler', StandardScaler())]),
                                 RangeIndex(start=0, stop=1212, step=1))])
RangeIndex(start=0, stop=1212, step=1)
SimpleMovingAverage()
StandardScaler()
LogisticRegression(max_iter=1000, penalty=None, random_state=42)
```

Figure 5: Example Pipeline for Logistic Regression (Baseline)

Support Vector Machines handle noisy data sets relatively easily by applying soft margins with a regularization parameter “c”. Soft Margin allows data points to fall on the wrong side of the decision boundary or margin. By allowing the noisy data points to fall on the wrong side, the model will prioritize the general trend of the data distribution by ignoring the made outliers. This comes with a trade-off of the regularization parameter “c”. Decreasing the “c” value will accept that some points will be misclassified, essentially ignoring the noise, leading to a more generalized model and reducing overfitting. The hyperparameter “kernel” will be set to ‘rbf’ due to the complexity of the nonlinear boundaries, and the hyperparameter ‘probability’ will be set to TRUE to output appropriate evaluation methods. Setting this hyperparameter to TRUE will add an additional pre-processing step known as Platt Scaling, where a separate logistic regression is done on top of the raw SVM output using 5-fold cross validation. It is important to note that the other parameters mentioned play a greater role in the tuning of the model.

For Decision Tree tuning, “max_depth” and “criterion” hyperparameters will go through cross validation. Max depth limits the depth of the tree. By limiting the depth of the tree, the decision tree will learn the general shape rather than getting convoluted by the noisy data. The criterion hyperparameter will be set to either ‘gini’ or ‘entropy’. The ‘gini’ calculation is much simpler to compute than entropy due to no logarithmic computation, leading to the assumption that ‘gini’ will be selected as the optimal hyperparameter.

With the same idea as the decision tree model, Random Forest hyperparameters max_depth and criterion will be cross validated (Figure 6). New hyperparameters that will be tuned are “n_estimators” and “max_features”. The “n_estimators” hyperparameter is the number of individual decision trees the model creates and combines before making a final decision. As the number of decision trees increases, the model captures a diverse range of patterns. This will increase the accuracy of the final prediction. However, there is a plateau that occurs when too many trees are created, which can lead to an increase in computational time. The “max_features” hyperparameter limits the number of features each individual tree can see before deciding on a split node. If a dominant feature is presented in every tree, then that feature will cause overfitting because that feature will be selected as the defining classifier. By implementing “max_features”, the model is forced to choose randomly from a smaller subset of each split, meaning that the dominant feature will not be available for all trees, only for some. This allows the model to not heavily rely on a dominant feature and generalize the model.



```
SimpleMovingAverage()
StandardScaler()
RandomForestClassifier(random_state=42)
```

Figure 6: Example Pipeline for Random Forest (Ensemble Method)

The hyperparameters that will be used for XGBoosting are like previous Decision Tree classification. Hyperparameters like “n_estimators” and “max_depth” will cycle through various values. The “objective” will be set to ‘binary:logistic’ because this is a standard binary classification. The “eval_metric” will be set to ‘logloss’. The learning rate is the step size for the gradient descent that controls how quickly the model updates during training to its minimum. If the learning rate is too large, the model will overshoot the optimal solution. If it is too small, then the computation time increases significantly, even though it will eventually find its optimal point.

Empirical Results and Evaluation

Classification evaluation metrics are a key component in defining the best model to use. Besides training and testing accuracy, it is important to dive deeper. Those two alone will not paint the full picture of how the model works. A confusion matrix is a table that summarizes the number of correct or incorrect predictions per class. This table will include true positives (positive samples were classified correctly), true negatives (negative samples were classified correctly), false negatives (negative samples were classified incorrectly), and false positives (positive samples were classified incorrectly). The goal is to maximize the true values (TP, TN). Precision score is the ratio of the true positives to all predicted positives. Recall score is the ratio of the true positives to all actual positives. Whether there is a need for a higher precision score over a recall score depends on the overall task. As mentioned earlier, devices that are classified as passing (hills) will be sent to other testing and packaging requirements. If any false negatives are missed, that will cause production delays and other malfunctions in systems. To avoid production delays, the goal is to minimize false negatives, meaning that the priority evaluation metric is having a higher recall score. However, too aggressive of a recall goal can cause too many false positives to go through, leading to taking up resources for actual good devices. The F1 score provides the harmonic mean between the precision score and recall score. Specificity is an evaluation metric that measures how well the model detects negative classes. This evaluation metric is also important for device classification problems because those devices that fail can go through R&D to identify common causes of device failure.

Tables 1-6 have the training accuracy, test accuracy, precision score, recall score, F1 score, specificity score and ROC AUC for each model discussed in the previous Methodology section. Note that some models did not go through extensive Column Transformations due to their inherent capability to bypass any need for transformations. Logistic Regression models are the only models that went through full untuned and tuned pipelines. Other models only went through one untuned pipeline, SMA + Standard column transformation as a baseline comparison, then tuning through grid search.

The best models based on output needs are selected for their training performance. The hyperparameters are presented in Table 7. The selected models were then passed through a test set evaluation, and their scores are presented in Table 8.

Based on the table and chart (Figure 7-11), the best overall model for pure classification is the "Tuned LR LOWESS + Standard" with its highest precision score, specificity score, and ROC AUC. This means that when the model predicts a "hill", it is 95% correct in its identification, and when the model predicts a "valley", it is 98.8% correct in its identification. However, the goal of this model is to classify true positives correctly in order not to delay production time. The "Tuned LR LOWESS + Standard model" has the lowest recall score of 0.2857. The model with the highest recall score is Tuned XGBoost with a recall score of 0.4887, meaning that the model has a 48.87% chance of identifying actual "hills". The F1 score is also the highest with 0.535, which shows a balance between the precision and recall score. Another parameter to think about is the computation time. LOWESS smoothing drastically increases computation time due to its iterative built-in linear regression. Finding the "Tuned LR LOWESS + Standard" hyperparameters took 1 hour and 15 minutes to compute. It is important to note that the cross validation parameters were limited due to limited computation time, so there is a possibility that these "best" models were not discovered.

Real-World/Social Impact

Although the dataset used for this model is simple and abstract, the classification of curves can be applied in many real world and social situations. For example, in medical diagnosis, the shape of an ECG curve can provide an early detection diagnosis of anomalies that can save a patient's life. Through the data collection of MRI or CT scans, the curvature of an organ can detect anomalies such as growth. The early detection can lead to providing minor invasive strategies to the patient to remove such found growth. Other societal impacts include detection of natural disasters. The pattern of seismic waves in a radar can help predict where an earthquake is imminent, which allows government officials ample time to plan accordingly (SUN, Deliang, CHEN, Danlu, MI, Changlin, CHEN, Xingyu, MI Shiwen, LI, Xiaoqin. n.d.).

Table 1: Results for Classification Performance of Untuned Logistic Regression Pipeline

Untuned Logistic Regression Pipeline	Training Acc. (%)	Testing Acc. (%)	Precision Score	Recall Score	F1 Score	Specificity Score	ROC AUC Score
SMA + Standard	62.95	55.14	0.6218	0.6152	0.6185	0.6431	0.7079
SMA + MinMax	63.67	54.32	0.6307	0.6173	0.6239	0.6552	0.7057
LOWESS + Standard	65.74	62.55	0.8543	0.3594	0.506	0.9415	0.8559
LOWESS + MinMax	61.61	58.85	0.8344	0.2664	0.4038	0.9496	0.8011

Table 2: Results for Classification Performance of Tuned Logistic Regression Pipeline

Tuned Logistic Regression Pipeline	Training Acc. (%)	Testing Acc. (%)	Precision Score	Recall Score	F1 Score	Specificity Score	ROC AUC Score
SMA + Standard	61.82	57.2	0.6833	0.4059	0.5093	0.8206	0.6832
SMA + MinMax	63.57	58.44	0.6840	0.4715	0.5582	0.7923	0.6975
LOWESS + Standard	74.72	72.43	0.9419	0.5137	0.6648	0.9698	0.9388
LOWESS + MinMax	73.27	68.31	0.9496	0.4778	0.6357	0.9758	0.9417

Table 3: Results for Classification Performance of Untuned/Tuned Support Vector Machine Pipeline

Untuned/Tuned SVM Pipeline	Trainin g Acc. (%)	Testing Acc. (%)	Precision Score	Recall Score	F1 Score	Specificity Score	ROC AUC Score
Untuned SMA + Standard	55.42	49.79	0.5631	0.3869	0.4586	0.7137	0.5552
SMA + Standard	65.63	56.38	0.7215	0.4820	0.5779	0.8226	0.7069
SMA + MinMax	65.53	56.38	0.7192	0.4820	0.5772	0.8206	0.7053
LOWESS + Standard	59.24	57.2	0.6806	0.3108	0.4267	0.8609	0.6235
LOWESS + MinMax	59.24	57.61	0.6806	0.3108	0.4267	0.8609	0.6236

Table 4: Results for Classification Performance of Untuned/Tuned Decision Tree Pipeline

Untuned/Tuned Decision Tree Pipeline	Trainin g Acc. (%)	Testing Acc. (%)	Precision Score	Recall Score	F1 Score	Specificity Score	ROC AUC Score
Untuned SMA + Standard	100	52.67	1.0	1.0	1.0	1.0	1.0
SMA + Standard	55.01	45.27	0.8033	0.1036	0.1835	0.9758	0.5625
SMA + MinMax	55.01	45.27	0.8033	0.1036	0.1835	0.9758	0.5625
LOWESS + Standard	55.01	45.27	0.8033	0.1036	0.1835	0.9758	0.5625
LOWESS + MinMax	55.01	45.27	0.8033	0.1036	0.1835	0.9758	0.5625

Table 5: Results for Classification Performance of Untuned/Tuned Random Forest Pipeline

Untuned/Tuned Random Forest Pipeline	Trainin g Acc. (%)	Testing Acc. (%)	Precision Score	Recal l Score	F1 Score	Specificity Score	ROC AUC Score
Untuned SMA + Standard	100	44.86	1.0	1.0	1.0	1.0	1.0
SMA + Standard	75.54	48.97	0.8333	0.6237	0.7134	0.881	0.8676
SMA + MinMax	75.54	48.97	0.8333	0.6237	0.7134	0.881	0.8676
LOWESS + Standard	75.54	48.97	0.8333	0.6237	0.7134	0.881	0.8676
LOWESS + MinMax	75.54	48.97	0.8333	0.6237	0.7134	0.881	0.8676

Table 6: Results for Classification Performance of Untuned/Tuned XGBoost Pipeline

Untuned/Tuned Random Forest Pipeline	Trainin g Acc. (%)	Testing Acc. (%)	Precision Score	Recall Score	F1 Score	Specificity Score	ROC AUC Score
Untuned	99.69	55.56	0.9979	0.9958	0.9928	0.9958	1.0
Tuned	100	53.5	1.0	1.0	1.0	1.0	1.0

Table 7: Results of Cross Validation Hyperparameters for the Best Models

Best Model(s) Hyperparameters	Hyper-Parameter 1	Hyper-Parameter 2	Hyper-Parameter 3	Hyper-Parameter 4	Hyper-Parameter 5
Tuned LR LOWESS + Standard	C (Regularization): 100	penalty: L2	solver: 'lbfgs'	Window Size: 5	N/A
Tuned SVM SMA + Standard	C (Regularization): 100	kernel: rbf	probability : True	Window Size: 5	N/A
Tuned RF SMA + Standard	criterion: 'gini'	n_estimators: 200	max_depth: 5	max_features: 'sqrt'	Window Size: 5
Tuned XGBoost	N/A	N/A	N/A	N/A	N/A

Table 8: Results for Classification Performance of the Best Model Pipeline

Best Model(s) Test Set	Testing Acc. (%)	Precision Score	Recall Score	F1 Score	Specificity Score	ROC AUC Score
Tuned LR LOWESS + Standard	60.08	0.95	0.2857	0.4393	0.988	0.8416
Tuned SVM SMA + Standard	56.38	0.6709	0.3985	0.5	0.7636	0.5619
Tuned RF SMA + Standard	48.97	0.5484	0.3835	0.4513	0.6182	0.5035
Tuned XGBoost	53.5	0.5909	0.4887	0.535	0.5909	0.5967

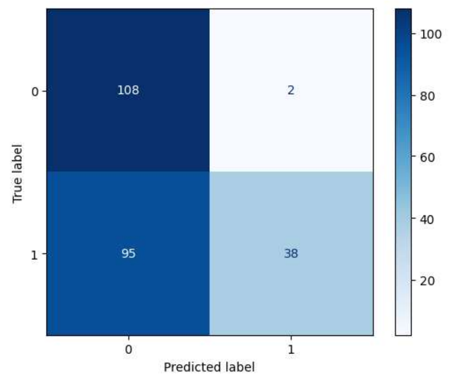


Figure 7: Confusion Matrix of Tuned LR LOWESS + Standard

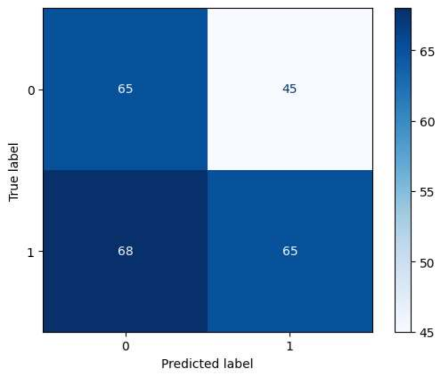


Figure 9: Confusion Matrix of Tuned XGBoost

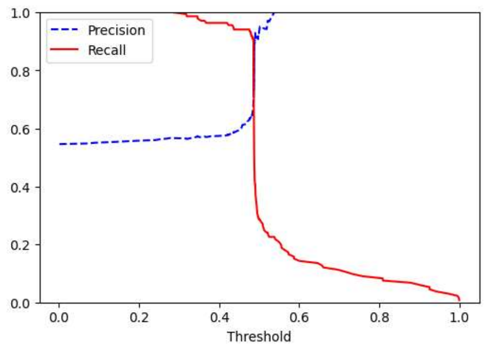


Figure 8: Precision-Recall Curve of Tuned LR LOWESS + Standard

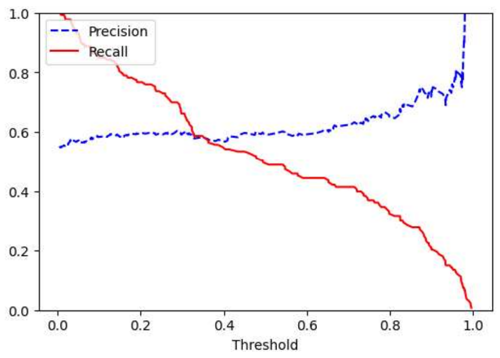
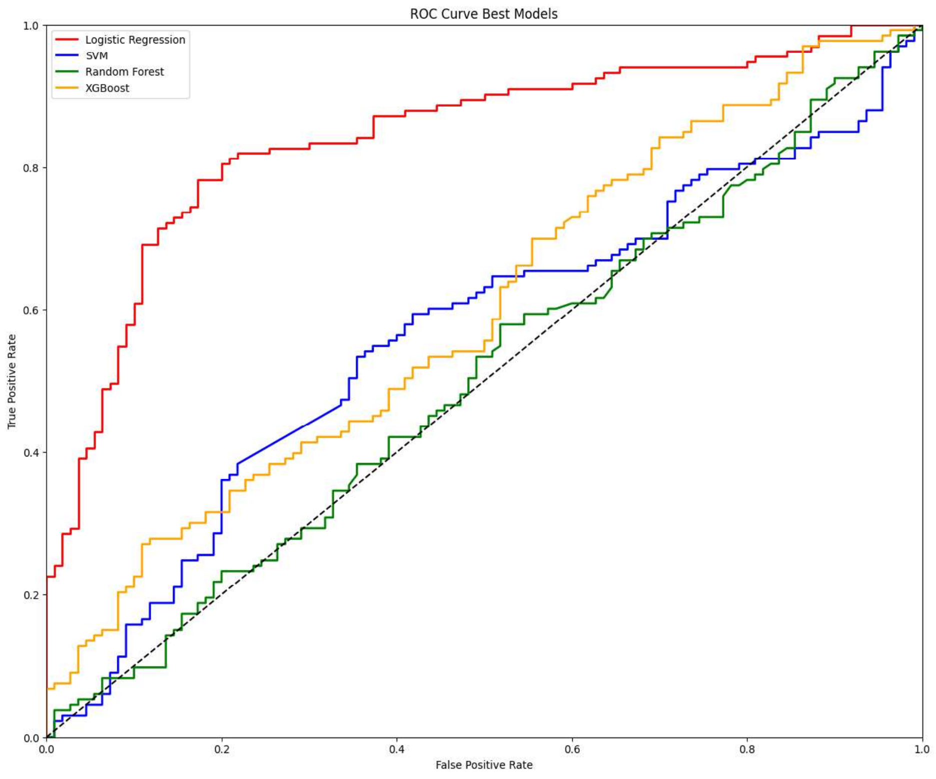


Figure 10: Precision-Recall Curve of Tuned XGBoost

Figure 11: ROC Curve Comparison of the Best Models



Based on the original motivation, it is important to hit benchmarks for the device output and increase efficiency with classification. In the beginning, a technician would have to look through tens of thousands of data sets and graphs and individually mark down which pad to pad curve passed parameters. Now, by using an ML model, the data can be parsed, transformed, and classified instantly, and identify which devices are ready for further testing.

The decision of the best model based on the company's evaluation metrics, however, brings an ethical dilemma. The ethical trade-off between the precision and recall score is one to be aware of, as is prominent in any machine learning model. For example, in the medical diagnosis world, having a model that scores high in precision and low in recall signifies that the model frequently makes false negative predictions. This means that the model can miss a true diagnosis, leading to the misdiagnosis of someone's life. In the case of the company's current benchmark goal, such as having a higher recall score, although it identifies good devices more accurately, it will increase the risk of false positive devices passing, and further delay testing time, creating a bottleneck effect (Theoretical Foundations from Ethical and Social Science Frameworks. n.d).

Lastly, there is an ethical dilemma of accountability and transparency of the models used. Although Logistic Regression and Decision Trees can be interpretable by the average person, models such as Random Forest and XGBoost are black box models, meaning that the model is very difficult to explain their decisions. A Logistic Regression model is interpretable based on the coefficients or weights of the hyperplane equation. The model's coefficients can be displayed in a bar chart to stakeholders to show which parameter holds much weight or decision power in its classification decision (Ethical AI: Addressing Bias and Fairness in Machine Learning Algorithms. 2024.). In a decision tree model, the decision tree can be constructed directly in scikit learn, showing how each division and decision are made step by step. Random Forest and XGBoost are much harder to visualize to stakeholders. If, for example, an XGBoost or Random Forest model was to predict whether someone is approved for an insurance claim and the classification output states no, the user cannot explain to the claimant how the decision was made. If there is no distinct explanation why a claimant could not get their insurance claim, that could lead to other altercations. In a company environment case, if the engineer presents these models to stakeholders in a TRB panel and is unable to explain why and how the model predicted its classifications, then the stakeholder will not provide support in time and resources for this model research to continue. There are current AI regulations and guidelines in the US, such as The Algorithmic Accountability Act and The Blueprint for an AI Bill of Rights, that provide an ethical framework on how to use AI appropriately (Ethical AI: Addressing Bias and Fairness in Machine Learning Algorithms. 2024.).

Conclusion/Future Work

A noisy dataset was provided with the task of classifying hills (1) and valleys (0). The dataset was then transformed through noise reduction techniques such as simple moving averages (SMA) or locally weighted scatterplot smoothing (LOWESS). The reduced noise dataset was then transformed through StandardScaler or MinMaxScaler techniques for models that require scaling, such as Logistic Regression and Support Vector Machines. The transformed data were tuned with respect to their model fit (Logistic Regression, Support Vector Machines, Decision Tree, Random Forest, and XGBoost) to find their optimal hyperparameters. A decision for the best model was made based on evaluation metrics such as having a higher recall and specificity score to meet the deliverables' needs. Based on the need to output working devices for further testing (higher recall), the best model for the goal is Tuned XGBoost. However, Tuned LOWESS + StandardScalar Logistic Regression had the highest test accuracy, precision score, and specificity score.

One personal challenge is the idea of scaling based on features rather than plotting. In other words, the scaling preprocessing step is to help optimize the regression models. The feature scaling goal is to standardize each feature (the column) because the models that require standardization will compare each index feature to each other to determine the weights of the feature. The weights of each feature will determine the method of classification. Comparing class 0 transformations (Figure 3-4), LOWESS + MinMax provides a clear valley dip at x = 40. With Standardization scaling, although the valley dip in x =40 is visible, the lowest point falls after x=80. This can get confusing for the average person when interpreting the graph. It is important to note that scaling is based on each column not row, meaning that the variance from any point in the graph is the same as each other and can ensure balanced input for the regression models. The act of scaling based on visual representation is called normalizing, where the goal is to bring all the samples to a unit length from each other (Chen, Yihao, Tang, Xin, Qi, Xianbiao, Li, Chun-Guang, Xiao, Rong n.d).

Another challenge is that LOWESS noise reduction takes a longer time to process than SMA. This is caused by the computational cost difference of LOWESS vs SMA. While SMA is, for lack of better words, simply averaging the neighbors, LOWESS is fitting a regression model at every point, causing an increase in computational time (National Institute of Standards and Technology, U.S. Department of Commerce. LOWESS.n.d).

For each model, there was a cross validation search for the best hyperparameters for each model to output their best evaluation metric. It took a considerably long time, especially for models that had LOWESS smoothing to go through cross validation search. The time to find the best parameter for LOWESS + Standard LR was 1 hour and 15 minutes. That is a considerably long time, especially for a model that was still limited in how many hyperparameters they could cycle through. It is assumed that, because I was unable to cycle through so many hyperparameters for each model, the best model is still out there.

I was unable to work with the company data due to their data being classified. I tried my best to find a dataset that was like the goal of classification based on the curvature of a graph. The Capacitance vs Voltage dataset is an exponential graph, and the preprocessing of the data before putting in the data into the machine learning model is much easier to compute. For example, any curves outside of the capacitance range are eliminated, or the capacitance caps and flatlines too early are immediately flagged as bad. Although the data can get noisy, it is not as noisy as the Hill Valley dataset. More often or not, the data that is being transformed and prepared for classification will not have any faulty curves.

For the future AI engineers who are tackling a classification problem based on the curvature of a graph, I recommend plotting the transformation technique to see if the transformations are working properly. It really helped me visualize what the data looked like and predict how the model will use the graph and its features to determine their predicted classification. I would also recommend a better computer, one that can process data with ease and reduce computation time. This will allow the engineer to try out more hyperparameters combinations that will bring better evaluation metric scores without the need to destroy your computer or wait hours upon hours for not so good results.

References

1. Brodley, C. E. 2008. Hill-Valley [Dataset]. UCI Machine Learning Repository. archive.ics.uci.edu (accessed December 8, 2025)
2. Radečić, Dario 2020. Data Scaling for Machine Learning – The Essential Guide. Towards Data Science. towardsdatascience.com/data-scaling-for-machine-learning-the-essential-guide-d6cfda3e3d6b/ (accessed December 6, 2025)
3. Chen, Yihao, Tang, Xin, Qi, Xianbiao, Li, Chun-Guang, Xiao, Rong . Learning graph normalization for graph neural networks. Neurocomputing. https://www.sciencedirect.com/science/article/abs/pii/S0925231222000030 (accessed December 6, 2025)
4. National Institute of Standards and Technology, U.S. Department of Commerce. LOESS (aka LOWESS). https://www.itl.nist.gov/div898/handbook/pmd/section1/pmd144.htm#:~:text=LOESS%2%200requires%20fairly%20large%2C%20densely,is%20a%20computational%20intensive%%2020method. (accessed December 6, 2025)
5. Lee, Fangfang. What is Logistic Regression? IBM. https://www.ibm.com/think/topics/logistic-regression#:~:text=Logistic%20regression%20i%20s%20a%20supervised,scikit%2Dlearn%20module%20in%20Python. (accessed December 5, 2025)
6. Kavlakoglu, Eda. What are SVMs? IBM. https://www.ibm.com/think/topics/support-vector-machine#:~:text=A%20support%20vector%20machine%20(SVM)%20is%20a,between%20each%20class%20in%20an%20N%2Ddimensional%20space. (accessed December 5, 2025)
7. Decision Trees. Scikit Learn. https://scikit-learn.org/stable/modules/tree.html (accessed December 5, 2025)
8. Kavlakoglu, Eda. What is Random Forest? IBM. https://www.ibm.com/think/topics/random-forest (accessed December 6, 2025)
9. RandomForestClassifier. Scikit Learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClass ifier.html (accessed December 8, 2025)
10. XGBoost. NVIDIA. https://www.nvidia.com/en-us/glossary/xgboost/ (accessed December 7, 2025)
11. SUN, Deliang, CHEN, Danlu, MI, Changlin, CHEN, Xingyu, MI Shiwen, LI, Xiaoqin. Evaluation of Landslide Susceptibility in the Gentle Hill-Valley Areas Based on the Interpretable Random Forest-Recursive Feature Elimination Model. SciEngine. https://www.sciengine.com/Jgeo/doi/10.12090/j.issn.1006-6616.2022128?trans=true#:~:text=In%20summary%2C%20the%20random%20forest,assessment%20and%20geological%20disaster%20prevention. (accessed December 8, 2025)
12. Theoretical Foundations from Ethical and Social Science Frameworks. National Academies. https://www.nationalacademies.org/read/26507/chapter/4 (accessed December 7, 2025)
13. Ethical AI: Addressing Bias and Fairness in Machine Learning Algorithms. 2024. Atlanta Technology Professionals. https://atpconnect.org/ethical-ai-addressing-bias-and-fairness-in-machine-learning-algorithms/#:~:text=In%20addition%20to%20those%20in,if%20the%20system%20behaves%20unexpectedly. (accessed December 5, 2025)
14. Mohamed, Rozlini. Yusof, Munirah Mohd. Wahid, Noorhaniza. 2018. A Comparative Study of Feature Selection Techniques for Bat Algorithm in Various Applications. ResearchGate https://www.researchgate.net/publication/323360911_A_Comparative_Study_of_Feature_Selection_Techniques_for_Bat_Algorithm_in_Various_Applications (accessed December 8, 2025)