Justin Morris
December 17 2025

RF Signal Classification

Project Introduction

Step 1

This project aims to train various machine learning models to classify arbitrary signals by their modulation type

Step 2

The dataset that will be used would be from kaggle dataset containing RF signal data.

Step 3

It Contains 164,160 signal samples, with features such as: Bandwidth, location, modulation type, device type, antenna type,CPU usage, memory usage and I/Q data,signal strength, power source, temperature ,etc.

Step 4

Of all these, the features I used to predict the modulation type were: frequency, signal strength, bandwidth, location device type, temperature, humidity, wind speed, precipitation interference type, power source, CPU usage, memory usage, WIFI strength, disk usage, system load.

Goal

Trained two machine learning models: a logistic regression model and a support vector machine. To Classify modulation types into 1 out 6 types: 8PSK, AM, BPSK, FM, QAM, QPSK

Logistic Regression

- Logistic regression was trained using a softmax classifier implemented with Scikit-learn's LogisticRegressionCV

- Categorical features(interference type, weather condition, device status, antenna type, device type, and modulation) were ordinally encoded.

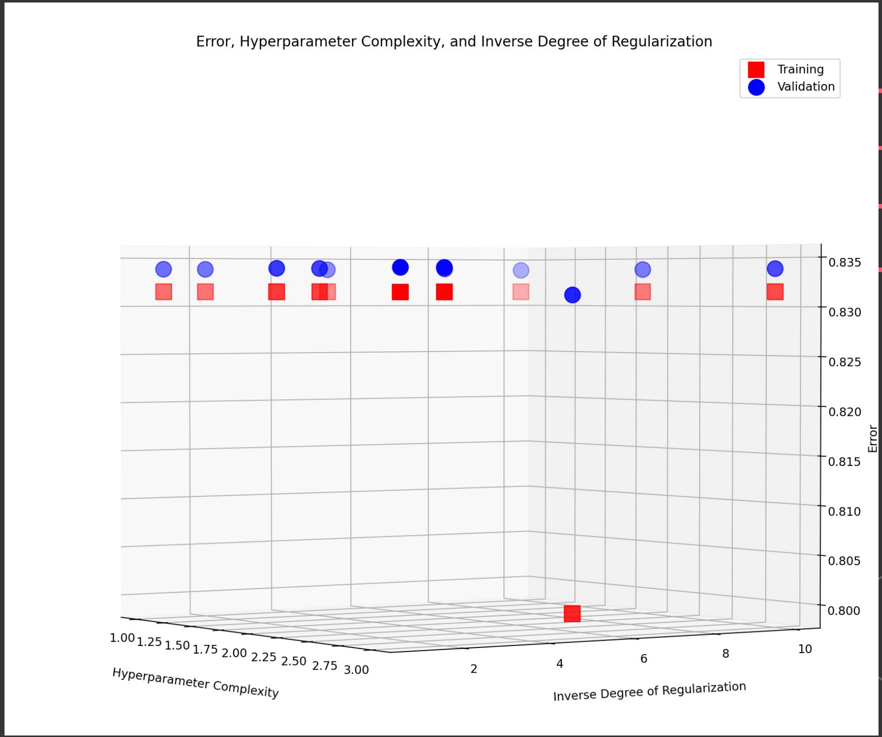
- All input features were standardized using StandardScaler, while class labels were kept separate.

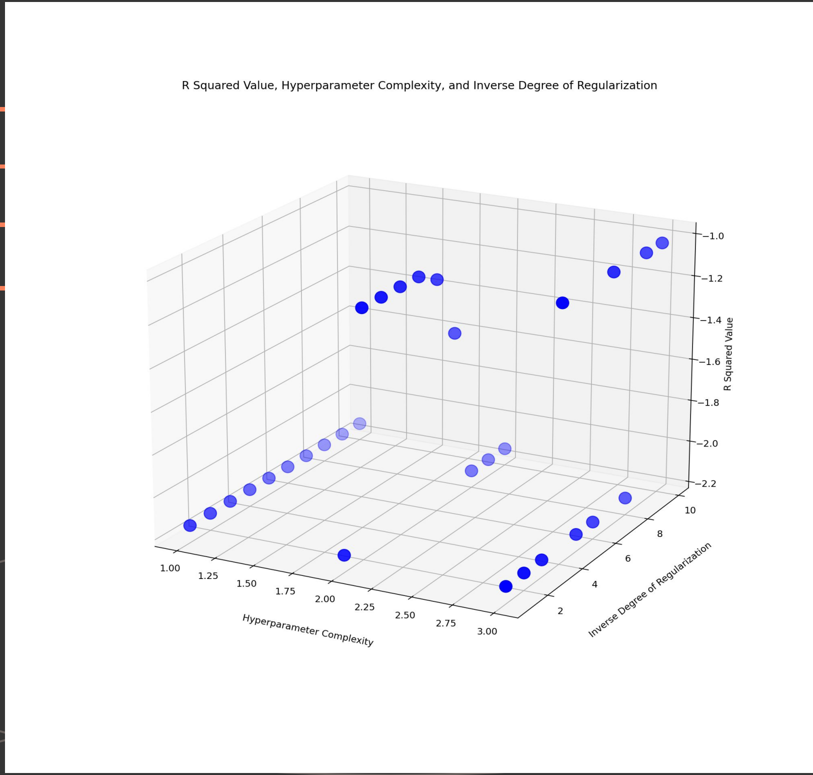
- The dataset was split into training and validation sets, with 25% reserved for validation

- LogisticRegressionCV enforces regularization, primarily L2.

- Regularization strength was controlled via the C parameter, tested over values from 1 to 10.

- Model performance was evaluated using stratified cross-validation, comparing tarined and validation error across hyperparameters.





Training and validation errors at varying of Cs and hyperparameter complexity.

R^2 values shown for the same values of Cs and the same hyperparameter complexities as inverse degree of regularization plot

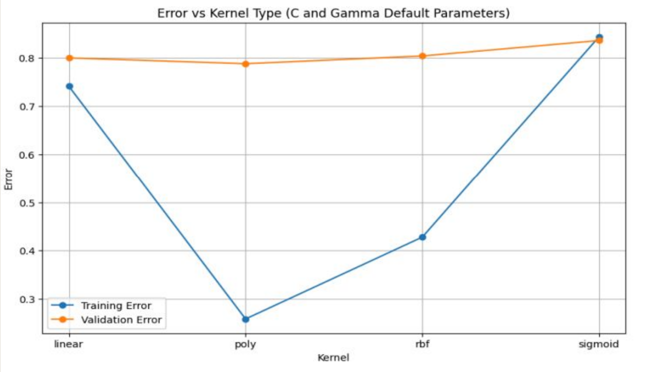
Support Vector Machine

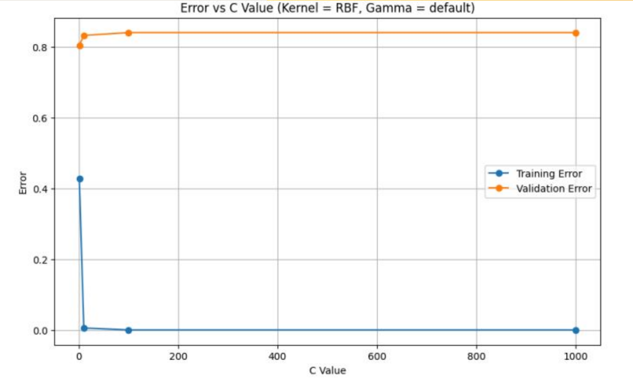
- To test different hyperparameters, we instantiated numerous support vector classification (SVC) objects derived from Scikit-Learn and experimented with the parameters one at a time.

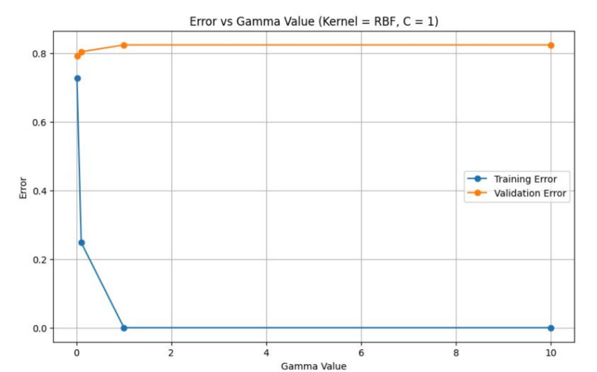
- Prior to instantiating SVC objects, we first performed a PCA feature transformation on the design matrix and reduced it to 10 dimensions. This would allow all SVC's to run faster.

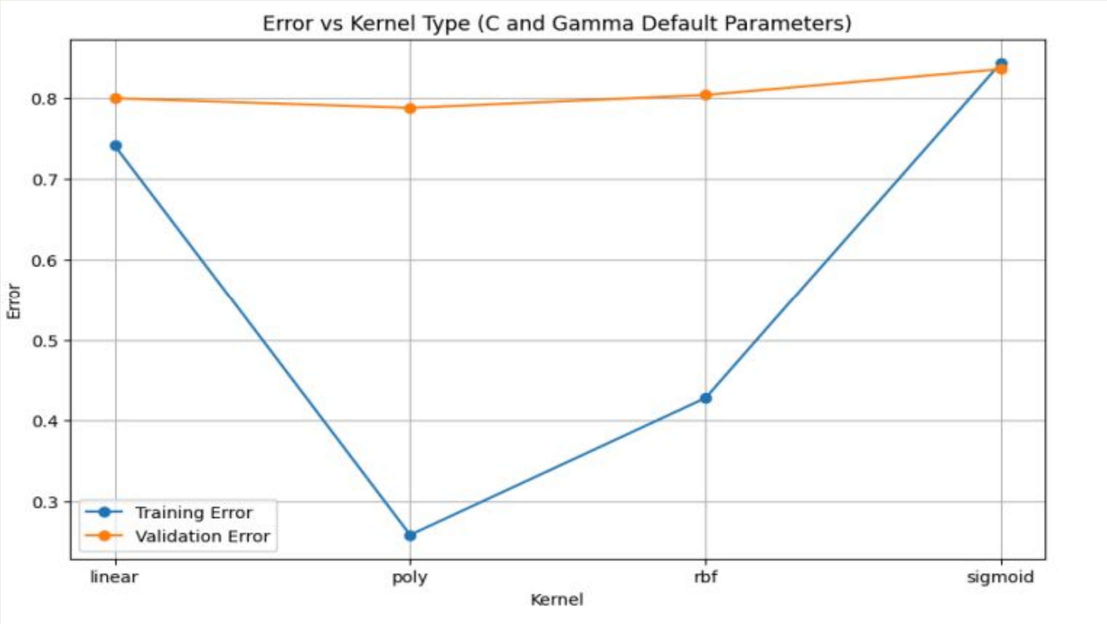
Support Vector Machine

The first parameter that was experimented with was the kernel type of the model. There were four kernel types that we compared: “Linear”, “Poly”, “RBF”, and “Sigmoid”. A plot displaying the training and validation errors as the parameters changed was generated as shown below





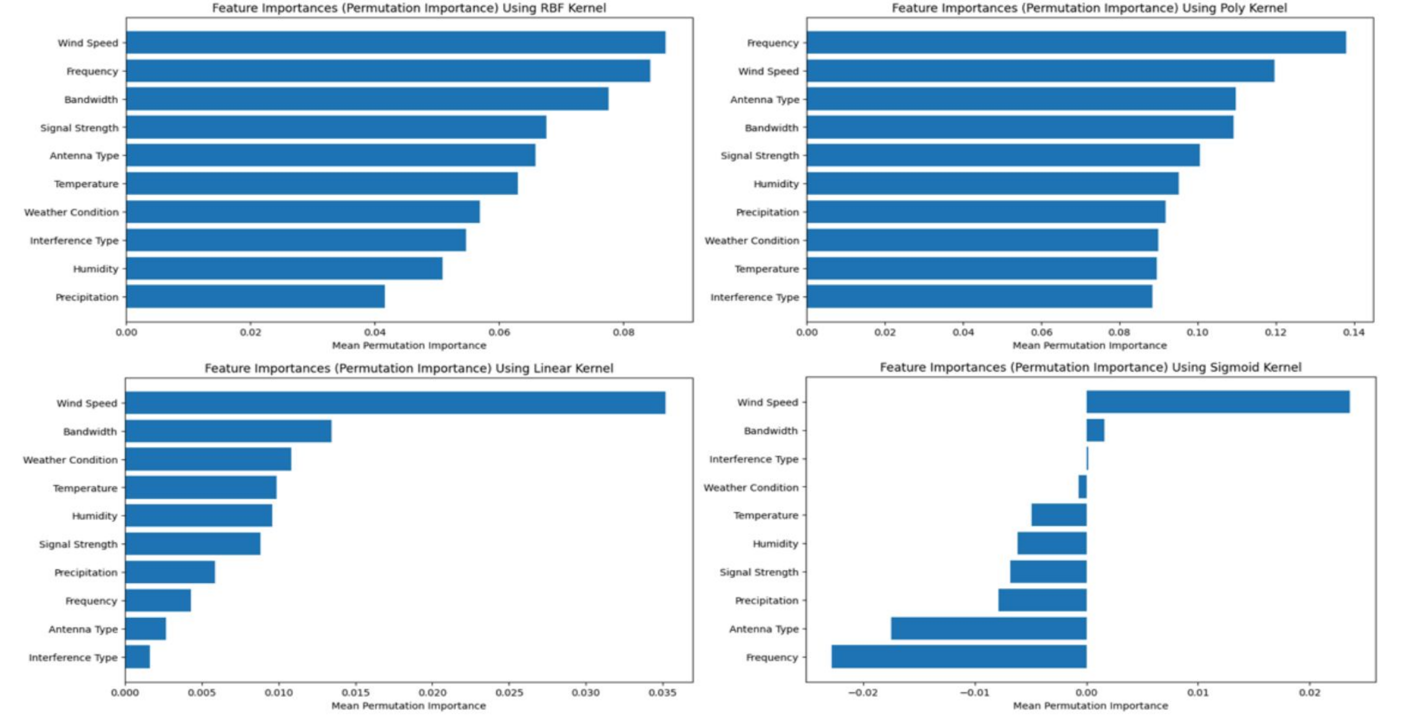


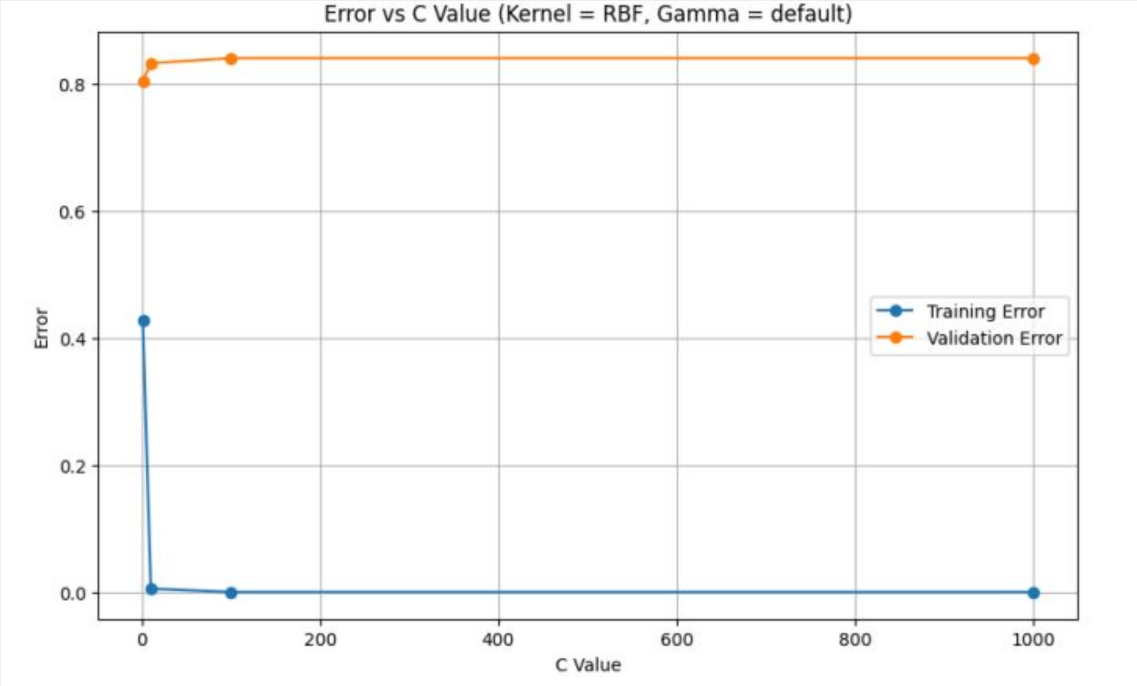


- The “linear” kernel type and the “sigmoid" kernel type caused the model to underfit which can be seen in their training and validation errors converging very closely. As for the model that used a “poly” kernel type, its variance was quite high given that its training error was way below its validation error compared to the other models.

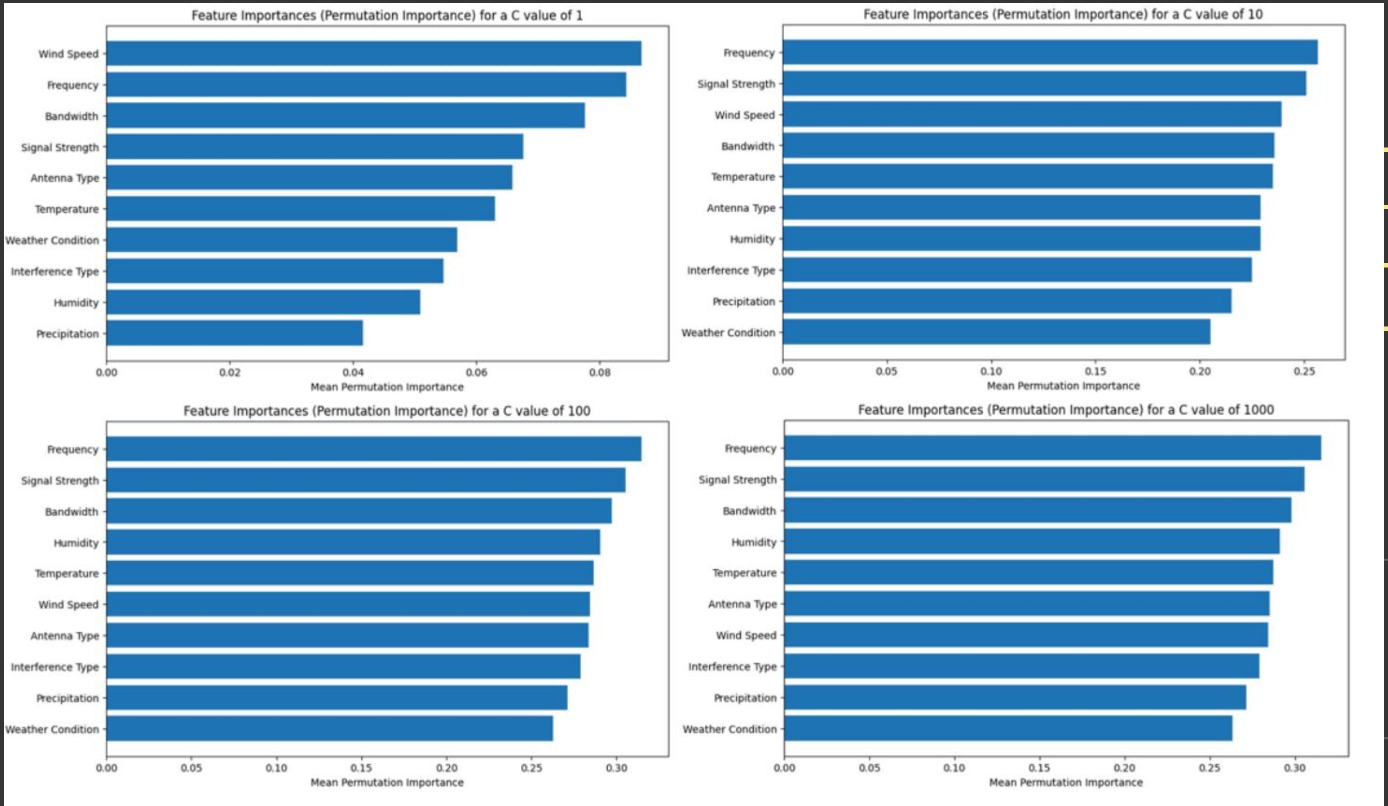
- This hints at overfitting occurring with the poly kernel type. Therefore, the “RBF” kernel type was chosen as the best one to fit this dataset best as a middle ground between overfitting and underfitting.

The weights were measured and visualized for their mean permutation importance when fitting the model. The permutation importance plots of 10 features for each kernel type were plotted as shown below.

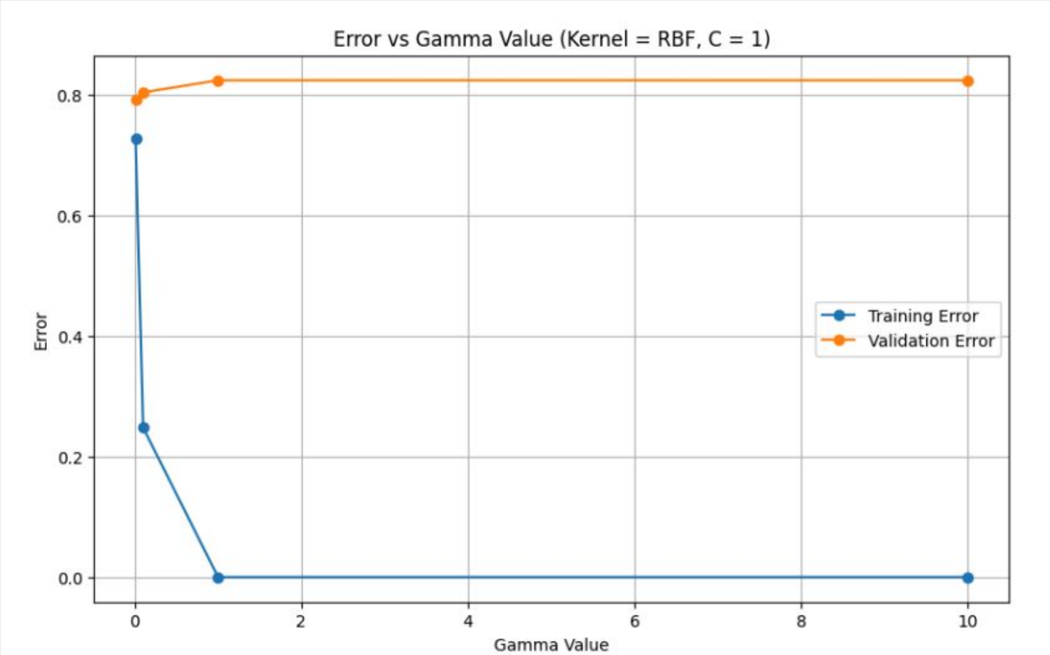




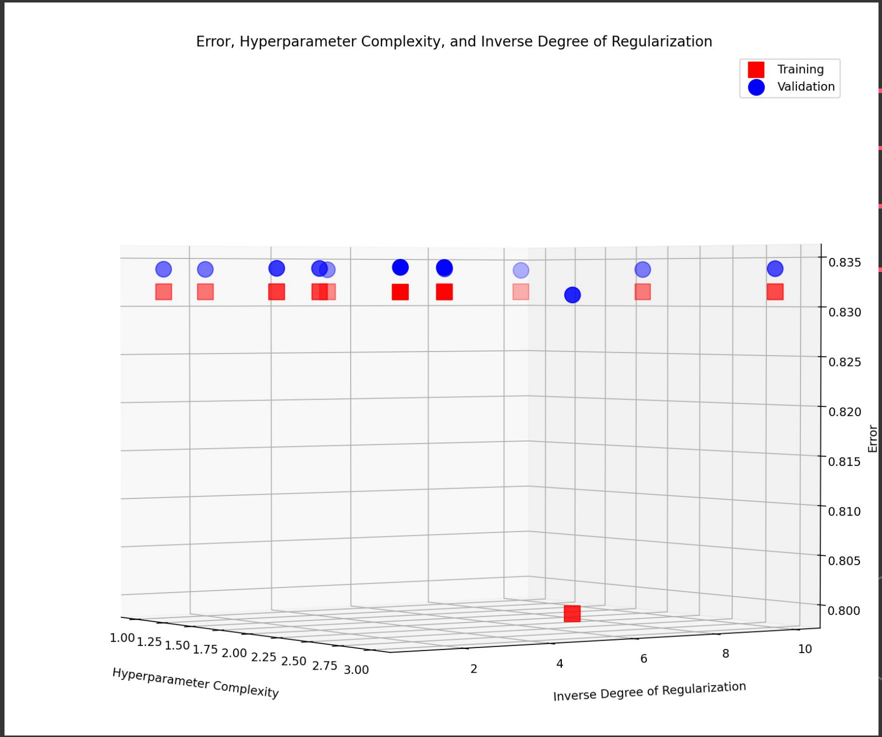
- The second parameter that was experimented with was the regularization parameter, C.
- The C values that were compared included 1,10,100, and 1000.
- These values were chosen to combat the underfitting that was visually prevalent based on the error plots from the kernel parameter experiment.

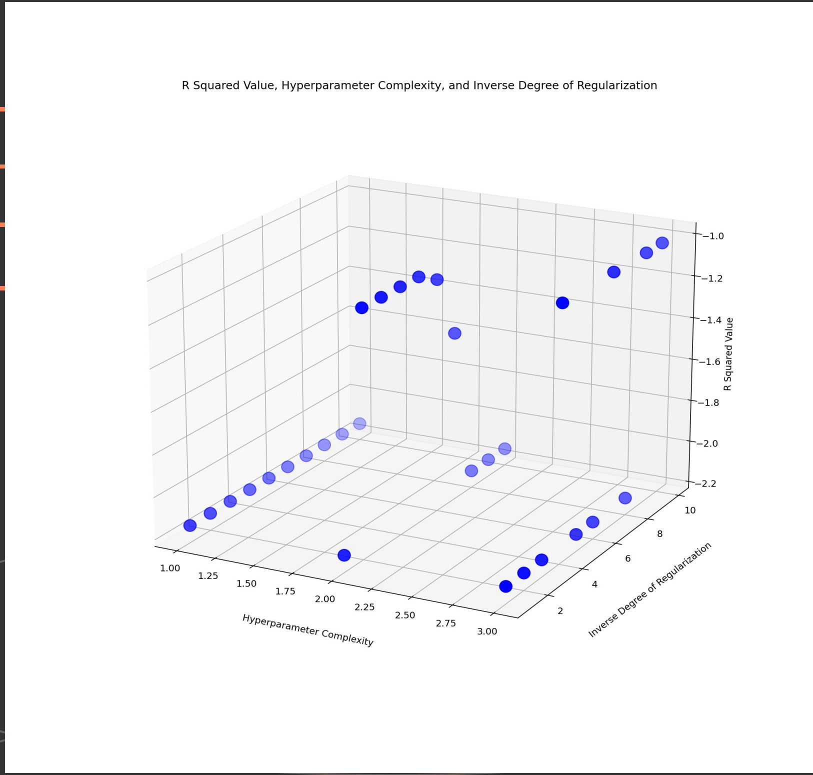


The error plot hinted, the higher the C value became, the more prevalent overfitting became. Thus, C value of 1 served as the best regularization parameter given that it did not overfit as extremely as the other C values. The permutation importance plots of 10 features for each C value were plotted as shown below



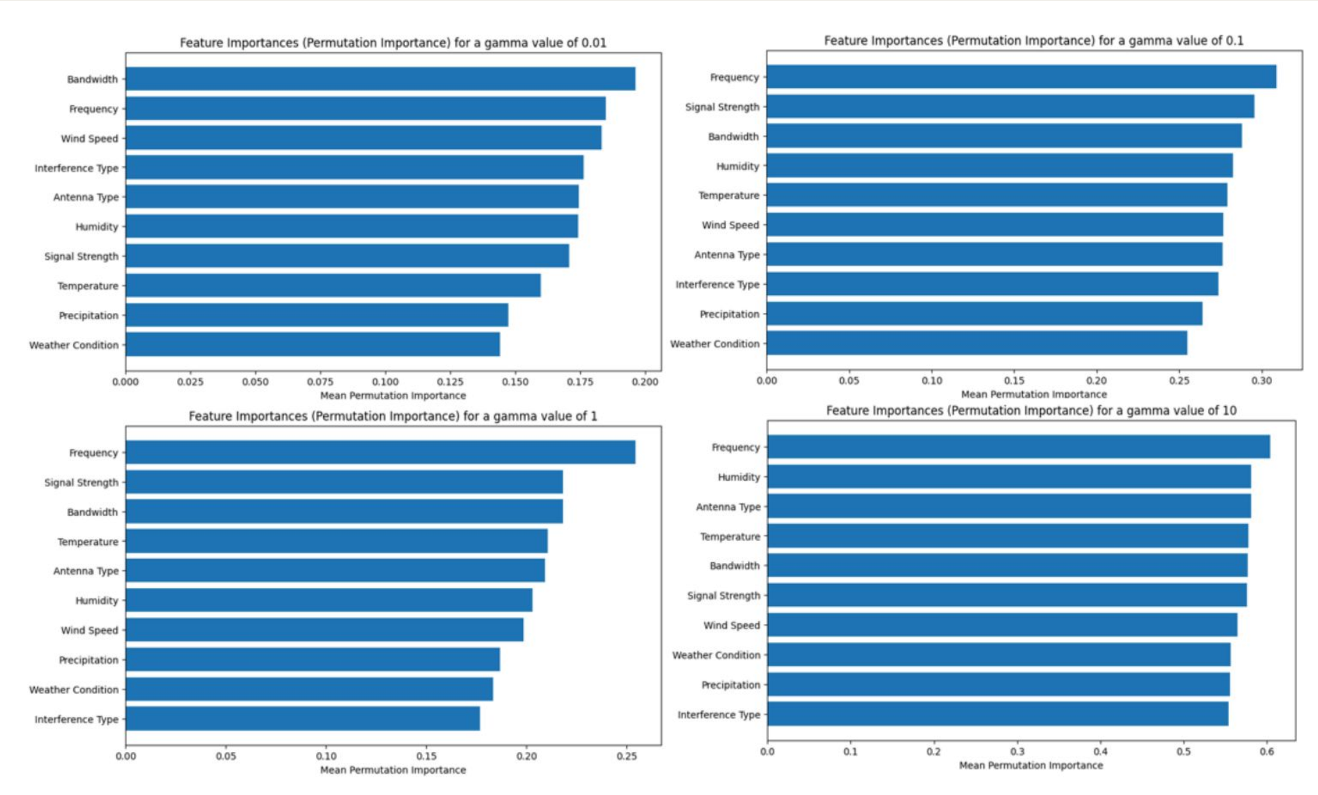
- The third feature that was experimented with was the gamma of the RBF kernel.
- This parameter is used to determine how complex the decision boundary of the SVM model will be. The gamma values that were compared included: 0.01, 0.1, 1, and 10. The training and validation error curves were plotted below as the gamma values changed.
- As gamma increased, the model seemed to exhibit much more overfitting. This is expected because increasing gamma increases the complexity of the decision boundary, and therefore we expect an increase in complexity to result in an increase of overfitting





Training and validation errors at varying of Cs and hyperparameter complexity.

R^2 values shown for the same values of Cs and the same hyperparameter complexities as inverse degree of regularization plot



As model complexity increases, a higher gap indicates excessive overfitting. The permutation importance plots of 10 features reinforces the idea.

Analytical Summary

Things observed

- Both models only marginally better than guessing.
- Hyperparameter tuning primarily led to overfitting or underfitting, not improved generalization.
- Logistic regression benefited slightly from increased model complexity, but validation accuracy remained low.
- SVM performance was sensitive to kernel choice and regularization.

Things to do differently

- A better dataset with more modulation specific features.
- Test More models and cross valid
- Focus on signal-domain feature before increasing model complexity.
- Improved features may enable meaningful gains with existing models.

Thank you