Classification Learner

The dataset that I chose for the classification learner is one that predict students' dropout and academic success. From a machine learning student's perspective, predicting student dropout and academic success using a classification learner dataset provides a comprehensive learning opportunity. It combines theoretical knowledge of algorithms with practical skills like preprocessing, feature selection, and model evaluation, while also emphasizing the societal impact and ethical considerations of machine learning applications. This makes it an ideal choice for my academic and personal growth in the field of machine learning.

As it relates to preprocessing steps taken, I started by converting the dataset into a CSV file, which serves as a standardized format for organizing the data. Then, I split the data into training and testing sets(70% and 30% respectively) to ensure an unbiased evaluation of model performance. . After importing the CSV into MATLAB, I perform data preprocessing steps: namely, handling missing values, encoding categorical variables, and scaling numerical features if needed. I train all classification algorithms, such as decision trees, Narrow Neural Network, and medium KNN, using the training set. During training, the model learns the patterns in the data by adjusting its parameters to minimize prediction error. Once trained, I evaluate the model on the test set, calculating metrics like accuracy, and precision. I also create performance visualizations such as confusion matrices, ROC curves, and learning curves to better understand the model's effectiveness. Finally, I record the output data from these evaluations, documenting the performance metrics to compare different algorithms and identify the best model for the classification task. My analysis and conclusions are further explained in this document:

Training results

This table below presents the Classification Learner training results for a dataset that predict Students' Dropout and Academic Success, comparing 33 algorithms based on their performance metrics. Below is a breakdown of what each column indicates:

Key Observations from the Table:

1. Highest Accuracy: The highest accuracy (76.9%) is achieved by the Linear SVM, followed closely by Bagged Trees (76.0%).
2. Lowest Error Rate: The Linear SVM also has the lowest error rate (23.1%), demonstrating its effectiveness.
3. Fastest Prediction Speed: The Narrow Neural Network has the highest prediction speed (51,000 obs/sec), making it the fastest for prediction.
4. Quickest Training Time: The Weighted KNN trains in the shortest time (1.3397 sec), followed by Coarse KNN (1.8573 sec).
5. Lowest Total Cost: The Bagged Trees algorithm has the lowest total cost (720), balancing both accuracy and cost-efficiency.
6. Poor Performers:

- Efficient Logistic Regression has the lowest accuracy (51.0%) and highest error rate (49.0%).
- Fine Gaussian SVM also struggles with low accuracy (53.3%) and high total cost (1402).

General Recommendations:

- For accuracy, using Linear SVM or Bagged Trees is recommended.
- For a balance of accuracy and speed, Narrow Neural Network or Bagged Trees are recommended.
- If computational resources are limited, using Weighted KNN, which trains the fastest is recommended.

Training results graph

Algorithm	Accuracy(%)	Total cost	Error rate (%)	Prediction speed(~obs/sec)	Training time (sec)
1. Fine tree	74.3	772	25.7	31000	27.996
2. Medium Tree	75.8	725	24.2	28000	11.057
3. Coarse Tree	71.9	842	28.1	30000	10.769
4. Linear Discriminant	75.8	725	24.2	23000	8.8942
5. Quadratic discriminant	71.7	849	28.3	25000	4.9794
6. Efficient logistic regression	51.0	1470	49.0	16000	26.84
7. Efficient Linear SVM	64.4	1067	35.6	15000	25.335
8. Gaussian Naive Bayes	67.9	963	32.1	31000	23.968
9. Kernel Naive Bayes	70.4	888	29.6	580	26.502
10. Linear SVM	76.9	694	23.1	23000	22.593
11. Quadratic SVM	75.8	727	24.2	18000	21.287
12. Cubic SVM	73.4	798	26.6	28000	19.896
13. Fine Gaussian SVM	53.3	1402	46.7	6200	17.455
14. Medium Gaussian SVM	75.6	731	24.4	6100	15.248
15. Coarse Gaussian SVM	74.7	760	25.3	7900	13.469

16. Fine KNN	65.5	1036	34.5	33000	31.019
17. Medium KNN	70.3	890	29.7	38000	3.4481
18. Coarse KNN	67.9	964	32.1	21000	1.8573
19. Cosine KNN	71.1	868	28.9	35000	4.4816
20. Cubic KNN	69.2	923	30.8	3500	7.3387
21. Weighted KNN	71.0	871	29.0	17000	1.3397
22. Boosted trees	75.8	726	24.2	18000	4.8039
23. Bagged trees	76.0	720	24.0	13000	10.159
24. Subspace disciminant	75.0	750	25.0	9000	7.7995
25. Subspace KNN	70.0	899	30.0	1700	12.451
26. RUSBoosted Trees	73.5	795	26.5	10000	11.064
27. Narrow Neural Network	73.7	790	26.3	51000	14.14
28. **Medium Neural Network**	67.6	971	32.4	92000	17.459
29. Wide Neural Network	70.9	872	29.1	14000	11.247
30. Bilayered neural Network	72.4	829	27.6	64000	15.434
31. Trilayered Neural Network	70.7	880	29.3	94000	17.437
32. SVM Kernel	74.7	758	25.3	26000	17.585
33. Logistics Regression Kernel	74.5	764	25.5	16000	12.393

Testing results

Testing Results Analysis

Key Observations from Testing Results:

1. Accuracy (%):
    - Highest Accuracy: The highest testing accuracy (74.9%) is achieved by Boosted Trees, followed closely by Medium Gaussian SVM (74.8%) and Bagged Trees (74.6%).

- Lowest Accuracy: Efficient Logistic Regression shows the lowest accuracy (47.7%), indicating poor performance during testing.
- Overall, accuracy values are slightly lower during testing compared to training, showing potential overfitting for some models.

2. Total Cost:
    - Lowest Total Cost: Boosted Trees has the lowest total cost (358), indicating the most cost-efficient performance during testing.
    - Highest Total Cost: Efficient Logistic Regression has the highest cost (745), correlating with its poor accuracy and high error rate.
3. Error Rate (%):
    - Lowest Error Rate: Boosted Trees has the lowest error rate (25.1%), making it the best performer for testing.
    - Highest Error Rate: Efficient Logistic Regression has the highest error rate (52.3%).
4. Overall Testing Trends:
    - Models such as Boosted Trees, Bagged Trees, Medium Gaussian SVM, and Linear SVM maintain consistently strong accuracy and cost-efficiency during testing.
    - Certain models like Efficient Logistic Regression and Fine Gaussian SVM struggle significantly with poor accuracy and high error rates.

Comparison Between Training and Testing Results

General Trends:

1. Accuracy Consistency:
    - For most algorithms, there is a small drop in accuracy during testing compared to training, indicating mild overfitting. However, the decrease is not drastic, showing that the models are reasonably well-generalized.
    - Top performers: Boosted Trees, Bagged Trees, Medium Gaussian SVM, and Linear SVM maintain strong performance in both training and testing.
    - Models like Efficient Logistic Regression and Fine Gaussian SVM perform poorly in both training and testing, reflecting intrinsic weaknesses.
2. Total Cost:
    - Testing total costs are significantly lower compared to training costs, possibly due to different misclassification penalties or testing data characteristics.
    - Boosted Trees consistently has the lowest total cost in both training (726) and testing (358), confirming its efficiency.

3. Error Rate:
    - Error rates follow similar trends as accuracy. The best algorithms (e.g., Boosted Trees) maintain low error rates during both training and testing.
4. Overfitting:
    - Models like Fine KNN and Wide Neural Network show relatively large accuracy drops during testing, suggesting potential overfitting during training.
    - Robust algorithms like Bagged Trees and Boosted Trees show minimal accuracy drops, reflecting better generalization to unseen data.

Summary of Training vs. Testing Results

Metric	Training Results	Testing Results	Observation
Highest Accuracy	Linear SVM (76.9%)	Boosted Trees (74.9%)	Testing accuracy is slightly lower than training, but top models remain strong.
Lowest Accuracy	Efficient Logistic Regression (51.0%)	Efficient Logistic Regression (47.7%)	Poor performance in both phases.
Best Overall Algorithm	Bagged Trees (low cost, high accuracy)	Boosted Trees (low cost, high accuracy)	Both are strong, well-generalized models.
Overfitting Observed	Fine KNN, Wide Neural Network	Fine KNN, Wide Neural Network	Significant accuracy drop from training to testing.
Consistent Performance	Bagged Trees, Boosted Trees	Bagged Trees, Boosted Trees	High accuracy and low cost in both phases.

Recommendations:

1. Best Overall Models:
    - Boosted Trees: Achieves the best performance in testing (accuracy: 74.9%, total cost: 358, error rate: 25.1%).
    - Bagged Trees: Consistently strong in both training and testing, with accuracy of 76.0% (training) and 74.6% (testing).
2. Avoid Poor Performers:
    - Efficient Logistic Regression and Fine Gaussian SVM consistently underperform in both training and testing.
3. Generalization:

- Models like Boosted Trees, Bagged Trees, and Linear SVM exhibit minimal accuracy drops between training and testing, indicating good generalization.

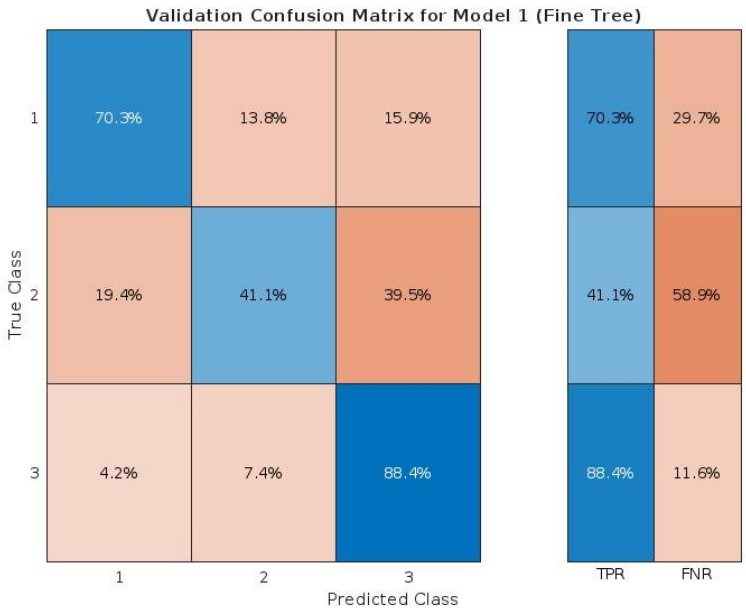
Testing results graph

Algorithm	Accuracy (%)	Total cost	Error rate (%)
1. Fine Tree	73.1	383	26.9
2. Medium Tree	74.0	370	26.0
3. Coarse Tree	67.7	461	32.4
4. Linear Discriminant	75.8	725	26.3
5. Quadratic discriminant	70.9	414	29.1
6. Efficient logistic regression	47.7	745	52.3
7. Efficient Linear SVM	65.3	494	34.7
8. Gaussian Naive Bayes	66.7	474	33.3
9. Kernel Naive Bayes	67.1	468	32.9
10. Linear SVM	74.6	362	25.4
11. Quadratic SVM	73.6	376	26.4
12. Cubic SVM	72.3	395	27.7
13. Fine Gaussian SVM	51.8	687	48.2
14. Medium Gaussian SVM	74.8	359	25.2
15. Coarse Gaussian SVM	73.4	370	26.6
16. Fine KNN	63.1	525	36.9
17. Medium KNN	69.5	434	30.5
18. Coarse KNN	65.5	491	34.5
19. Cosine KNN	70.5	420	29.5
20. Cubic KNN	68.4	450	31.6
21. Weighted KNN	69.9	428	30.1
22. Boosted trees	74.9	358	25.1
23. Bagged trees	74.6	362	25.4
24. Subspace discriminant	73.6	376	26.4

25. Subspace KNN	67.6	461	32.4
26. RUSBoosted Trees	71.4	407	28.6
27. Narrow Neural Network	72.5	391	27.5
28. Medium Neural Network	68.3	451	31.7
29. Wide Neural Network	68.2	453	31.8
30. Bilayered neural Network	70.8	416	29.2
31. Trilayered Neural Network	71.8	402	28.2
32. SVM Kernel	72.7	389	27.3
33. Logistics Regression Kernel	73.2	381	26.8

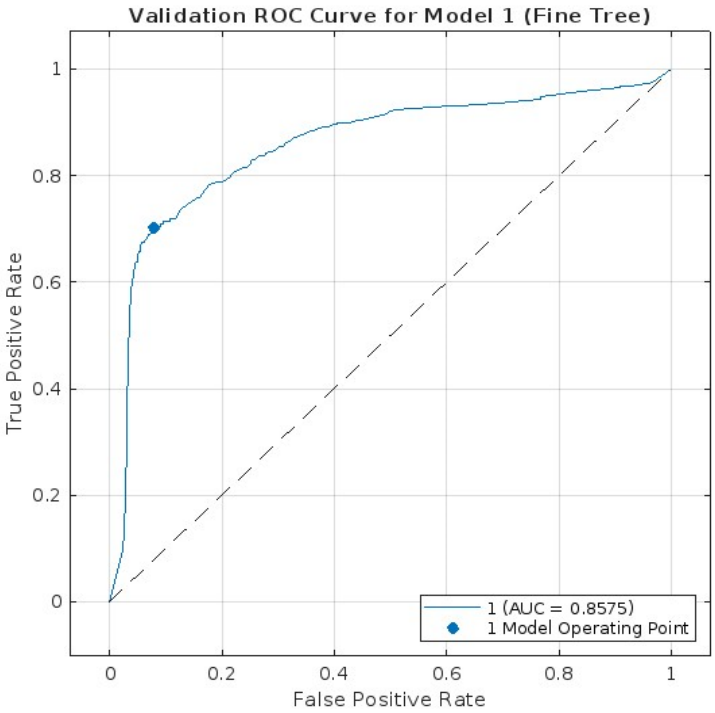
Graphs

Fine Tree - Training graphs



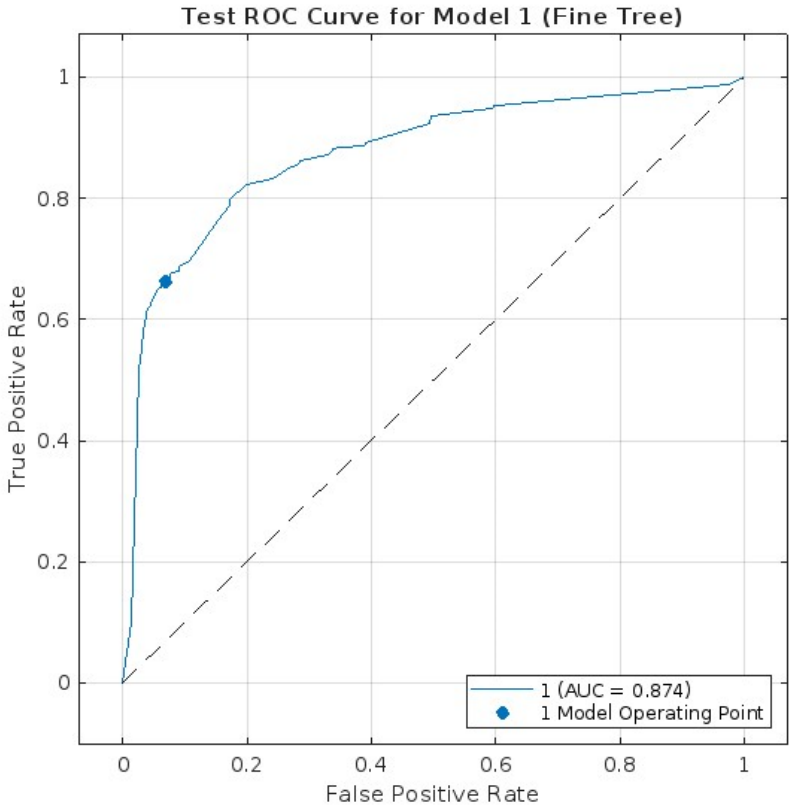
The confusion matrix shows the Fine Tree model performs best on **Class 3** with a True Positive Rate (TPR) of **88.4%** and low misclassification. **Class 1** is moderately accurate with

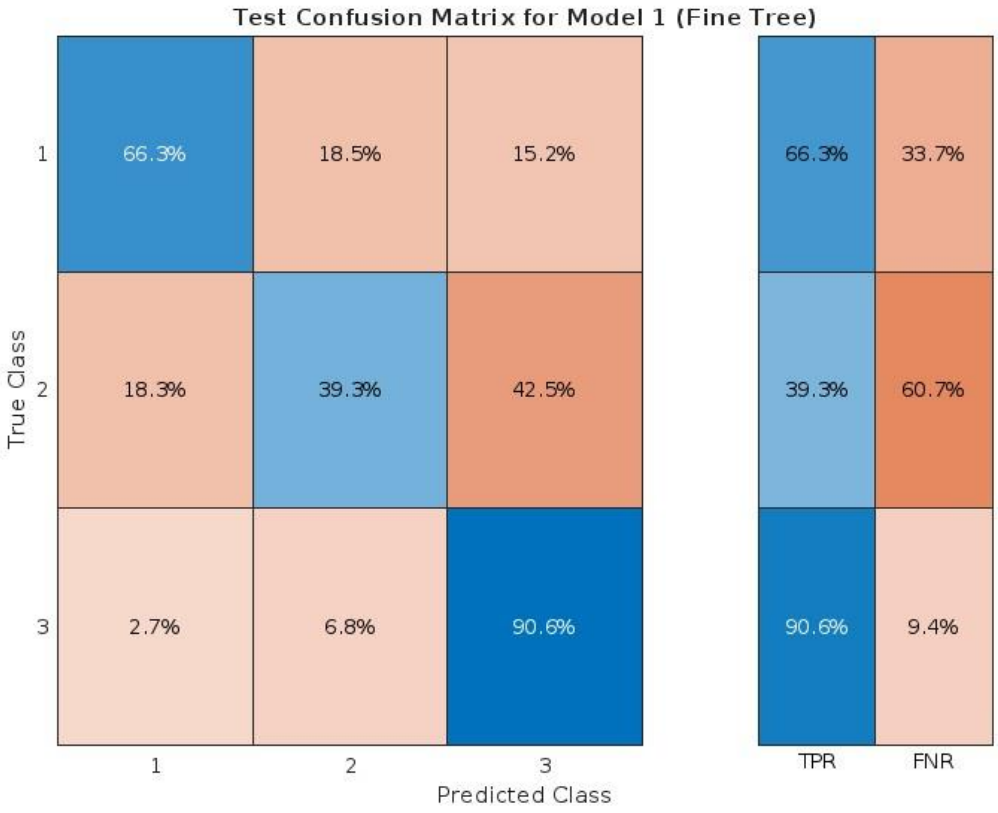
a TPR of **70.3%**, while the model struggles with **Class 2**, achieving only **41.1%** TPR and high confusion with Class 3 (**39.5% misclassified**).



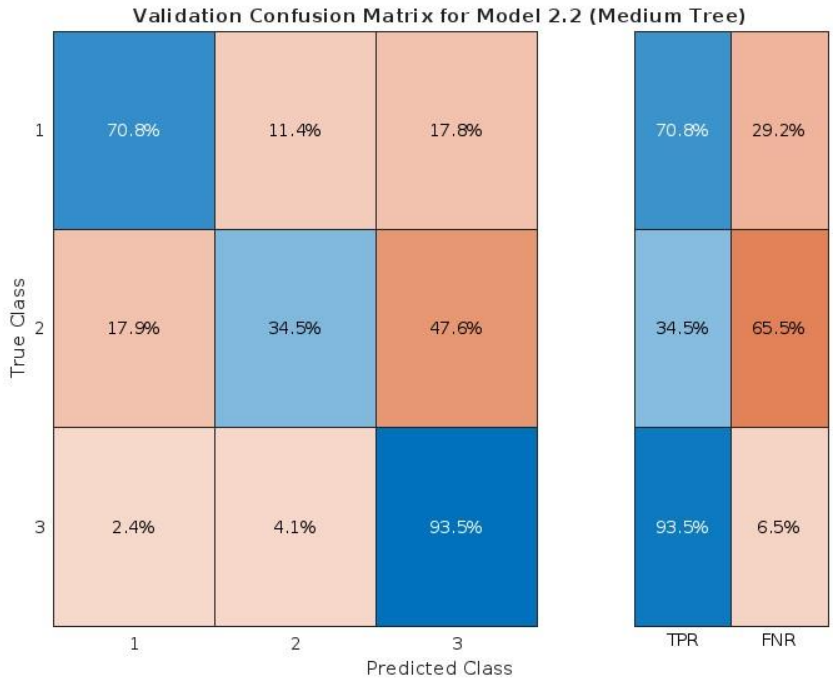
The graph shows the trade-off between the True Positive Rate (sensitivity) and the False Positive Rate for different classification thresholds. The curve's Area Under the Curve (AUC) is 0.8575, indicating good model performance. The dashed diagonal line represents random guessing, and the ROC curve's significant rise above this line demonstrates the model's ability to distinguish between classes effectively. The model's operating point is marked, reflecting its performance at a specific threshold.

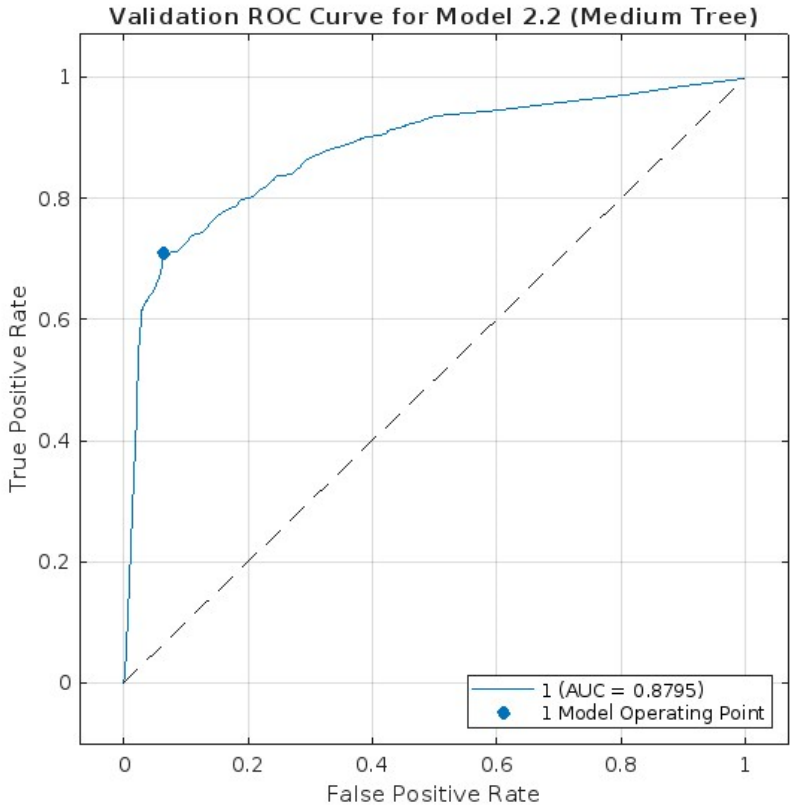
Testing Graphs



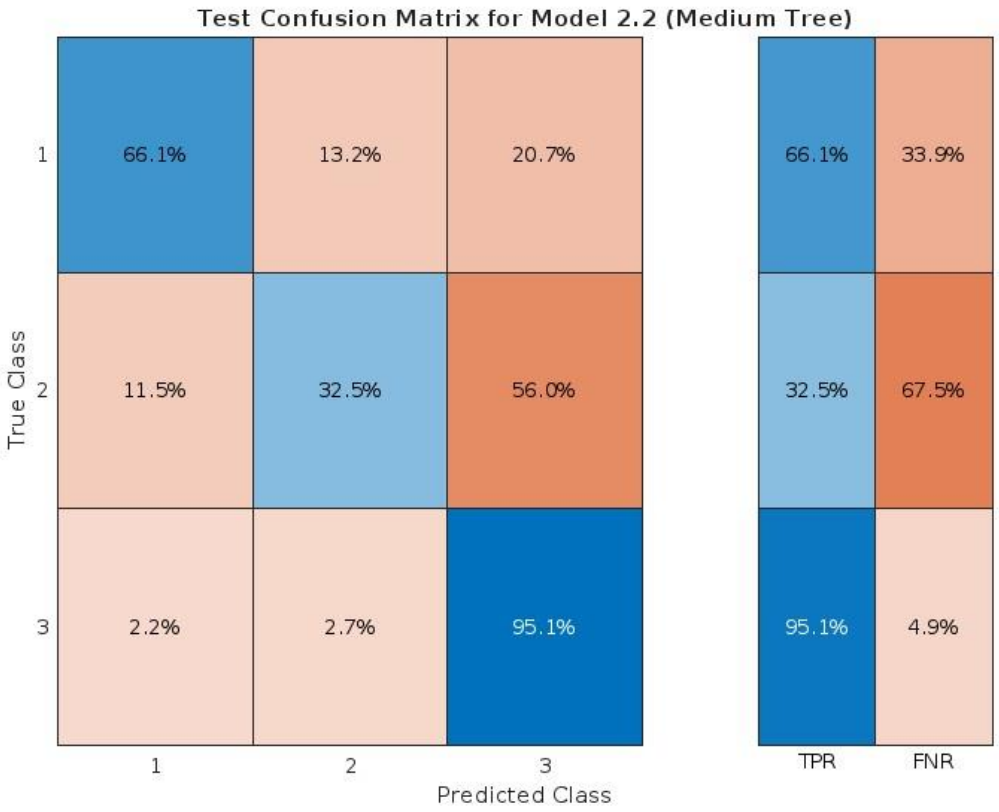


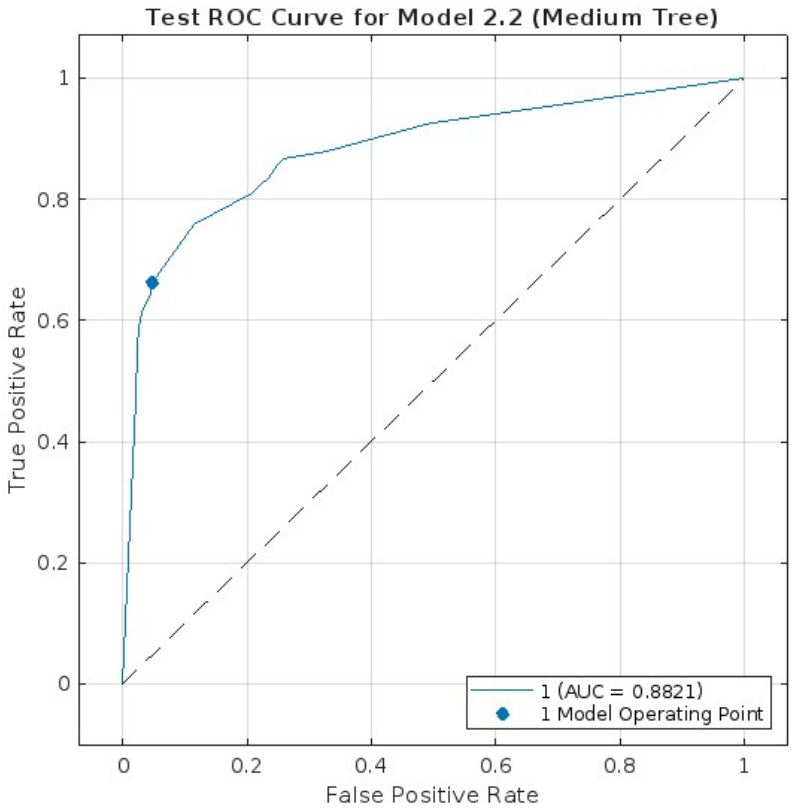
Medium Tree- Training graphs



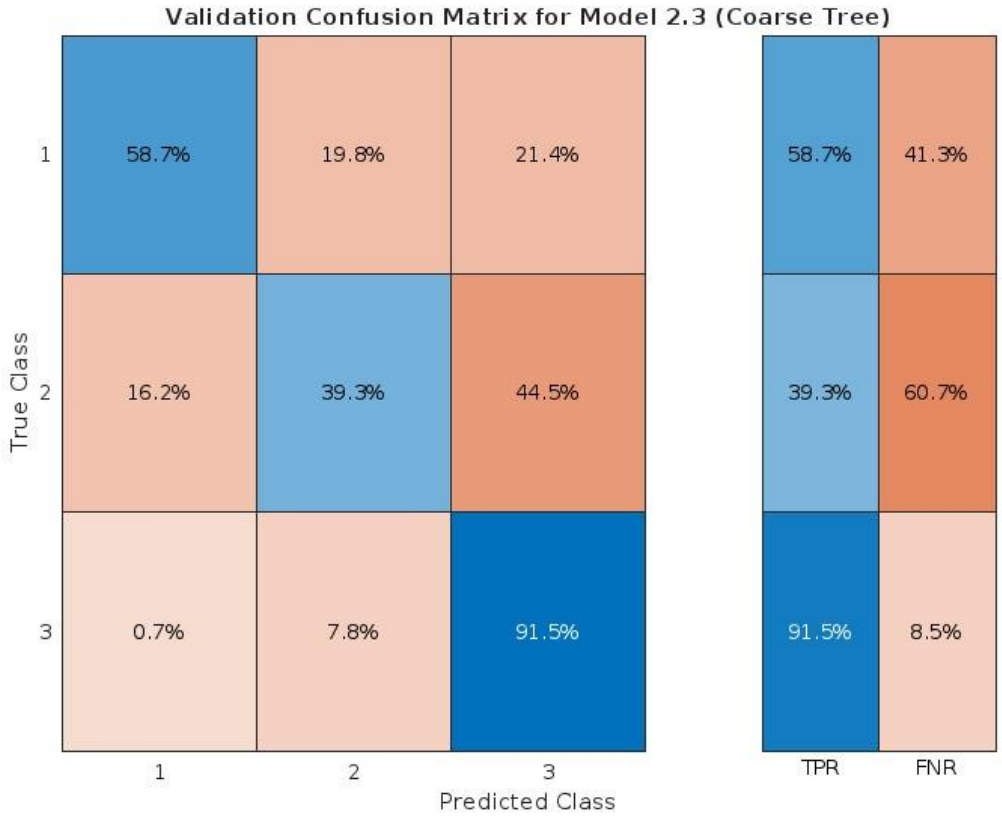


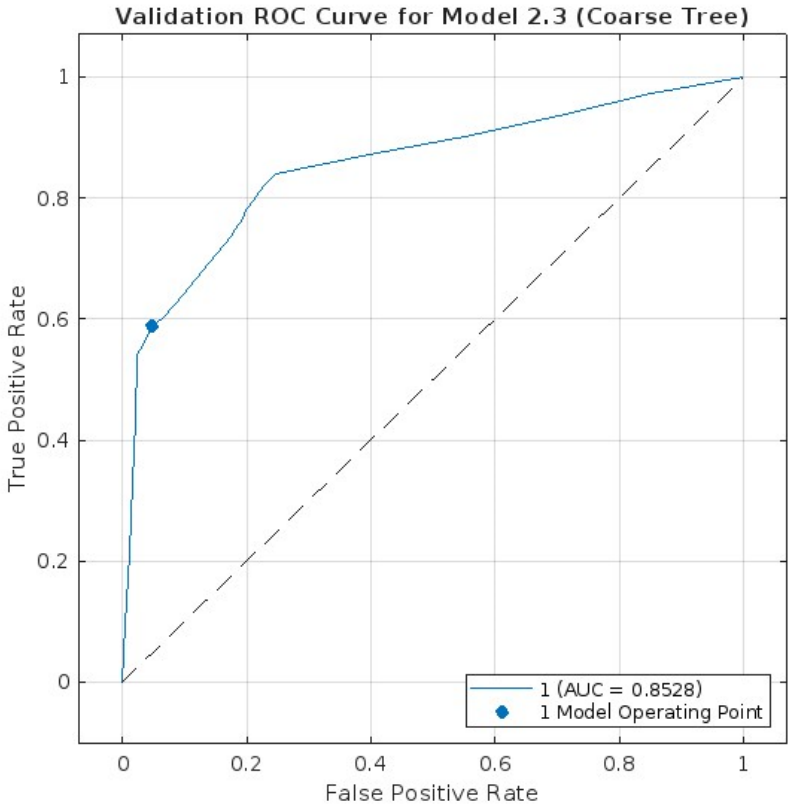
Testing Graphs



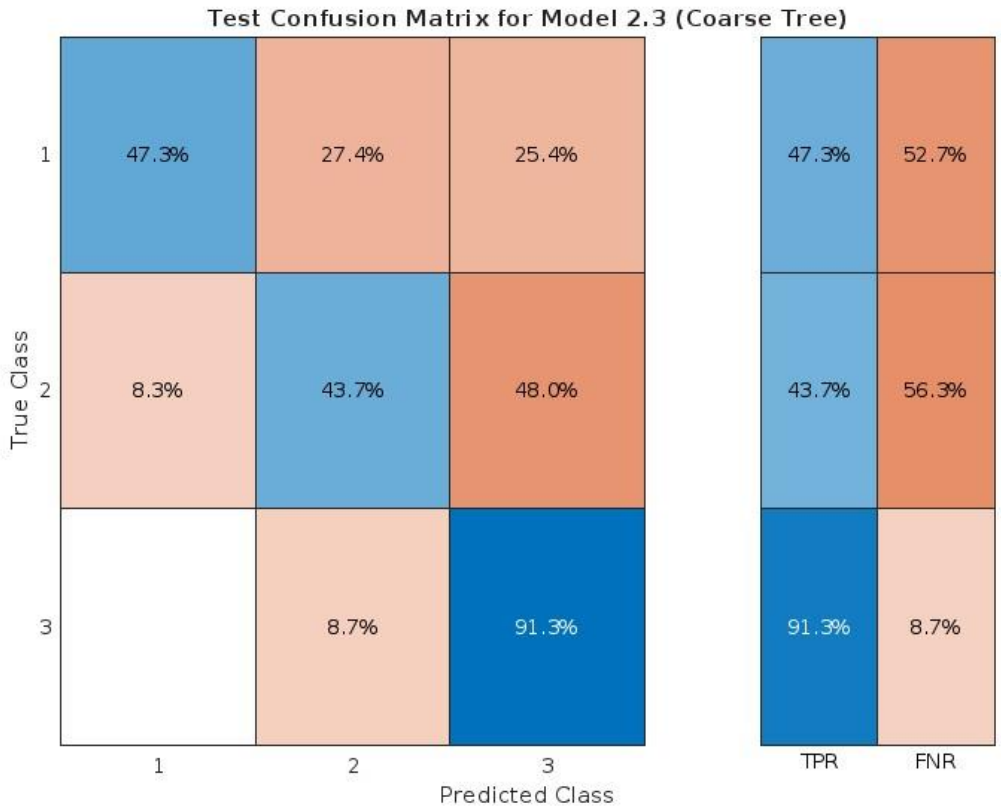


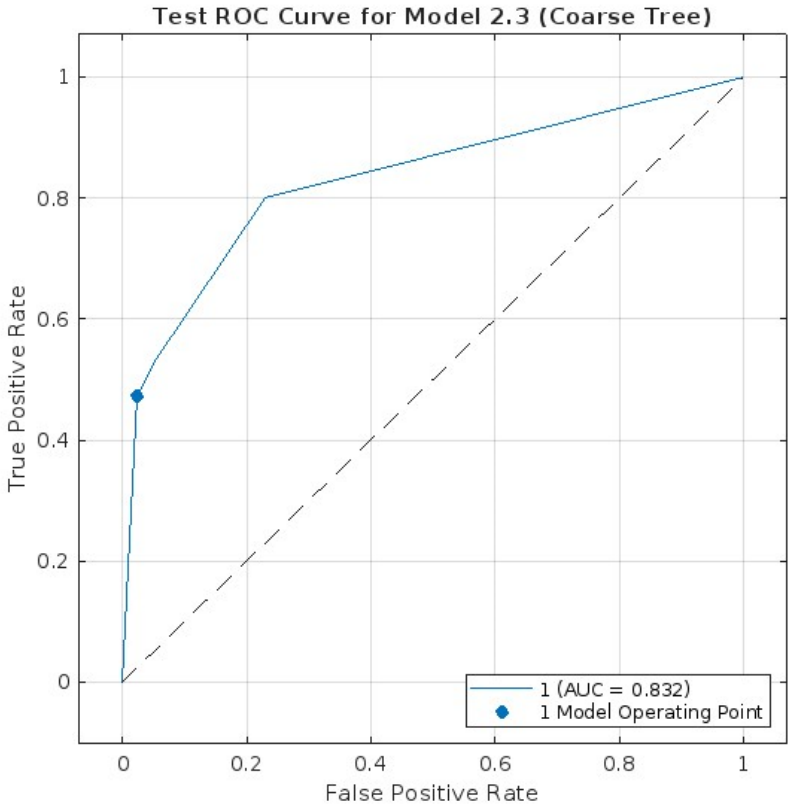
Coarse Tree – Training graphs



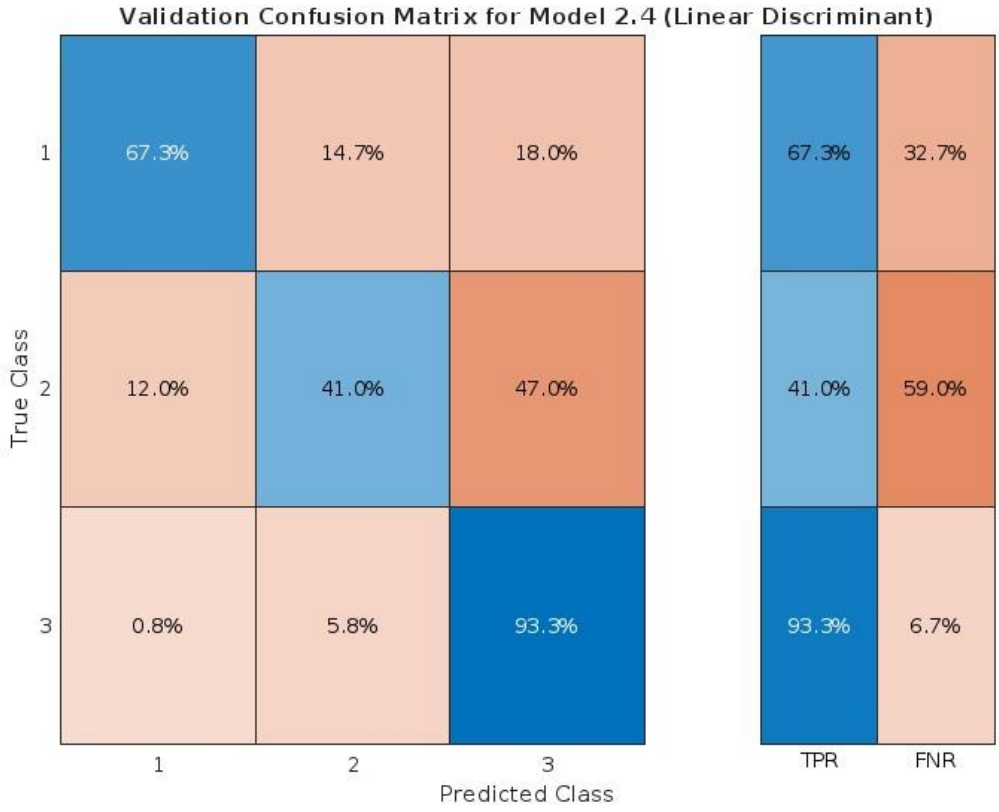


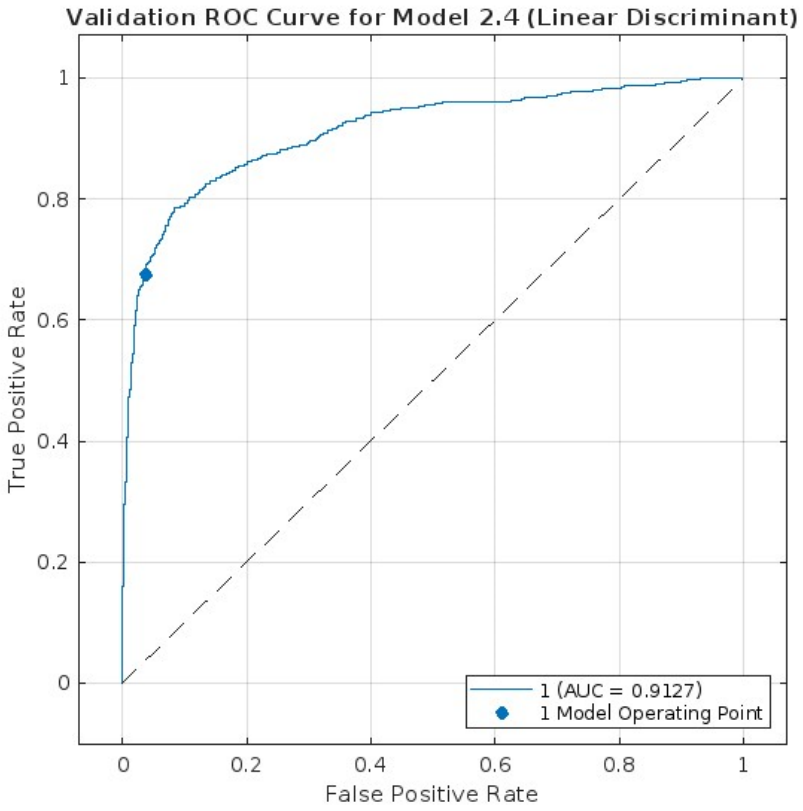
Testing graphs



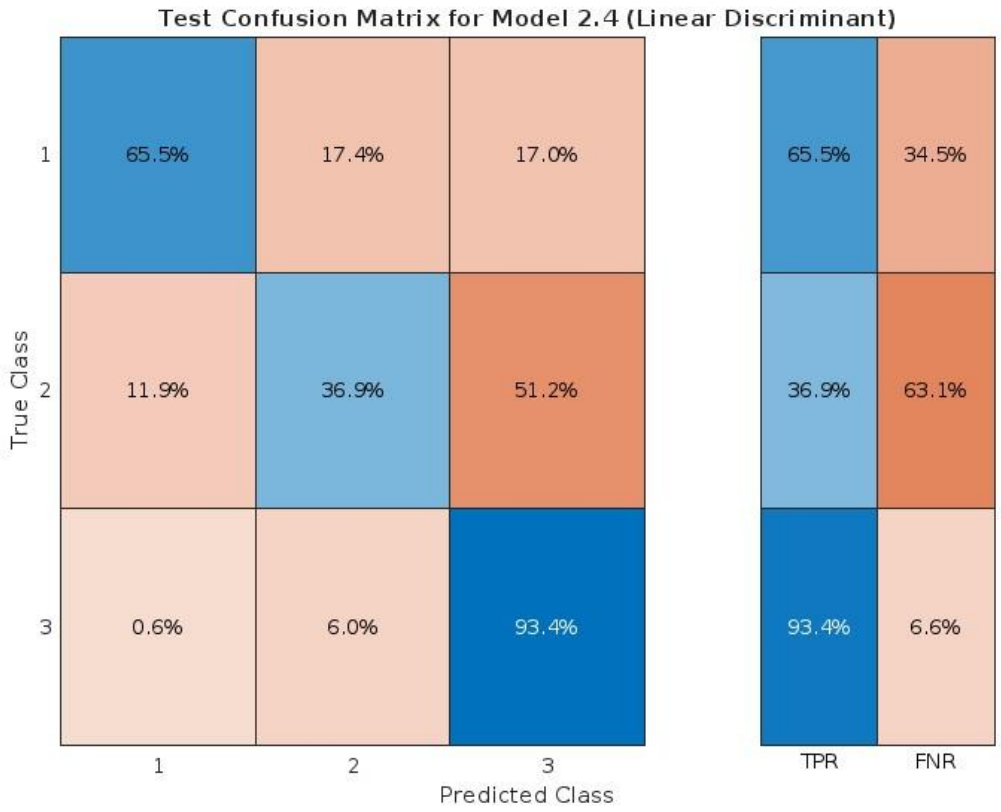


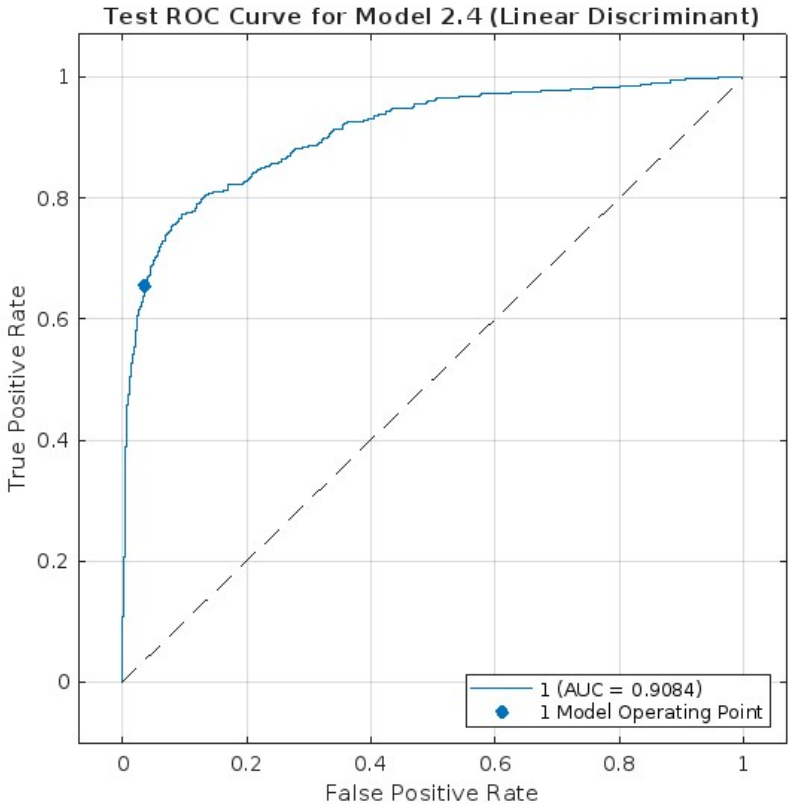
Linear Discriminant



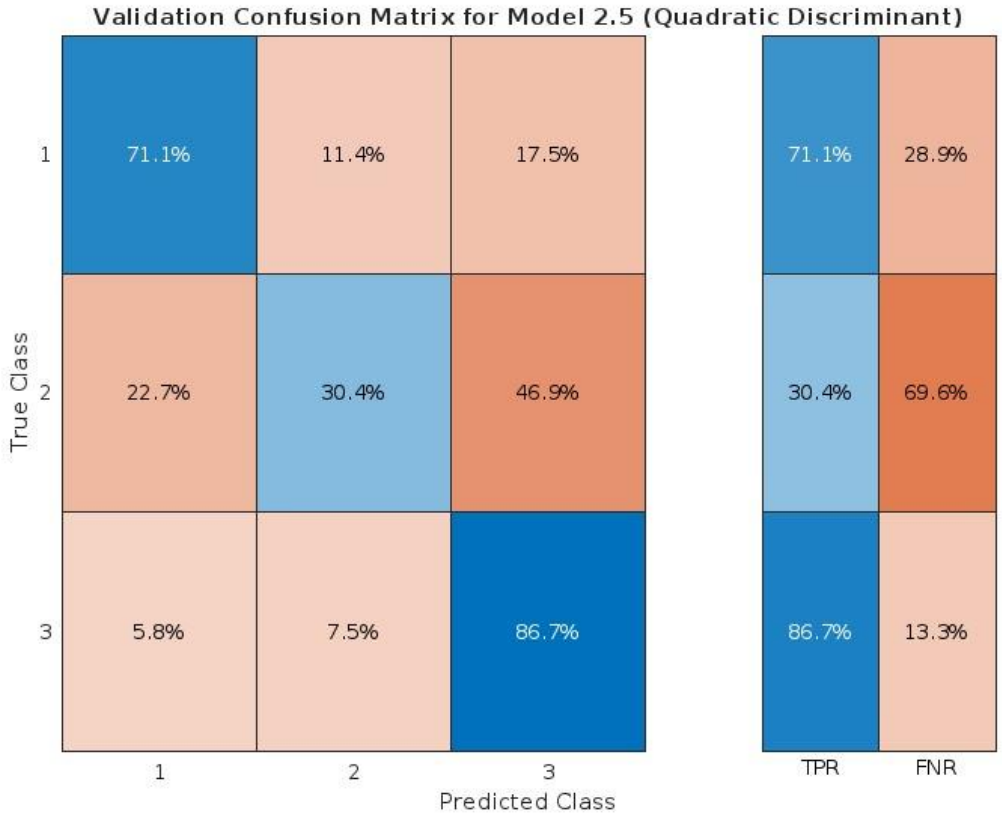


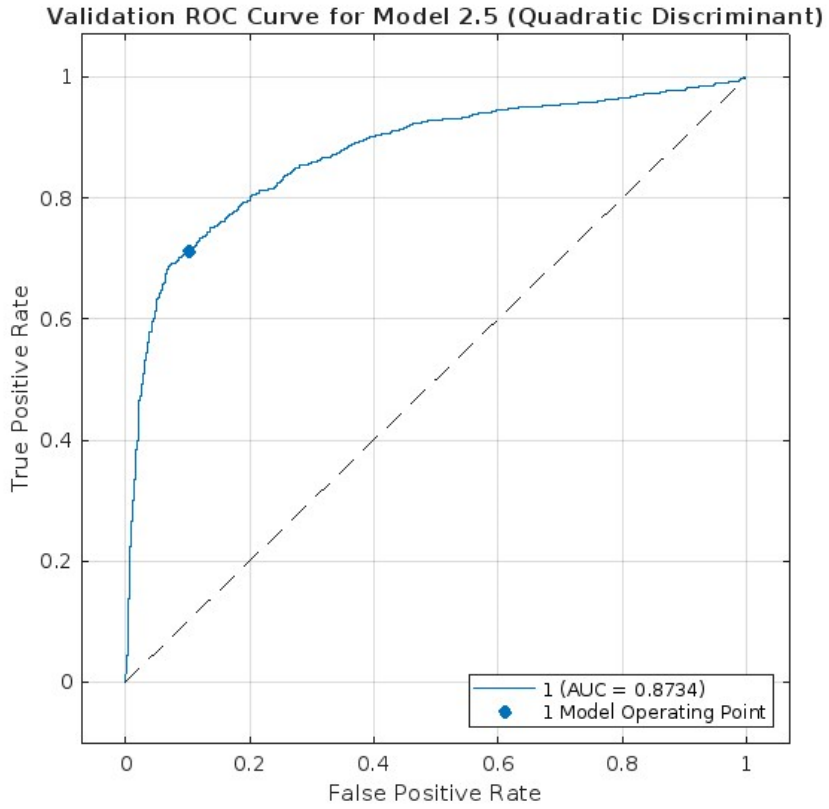
Testing graphs



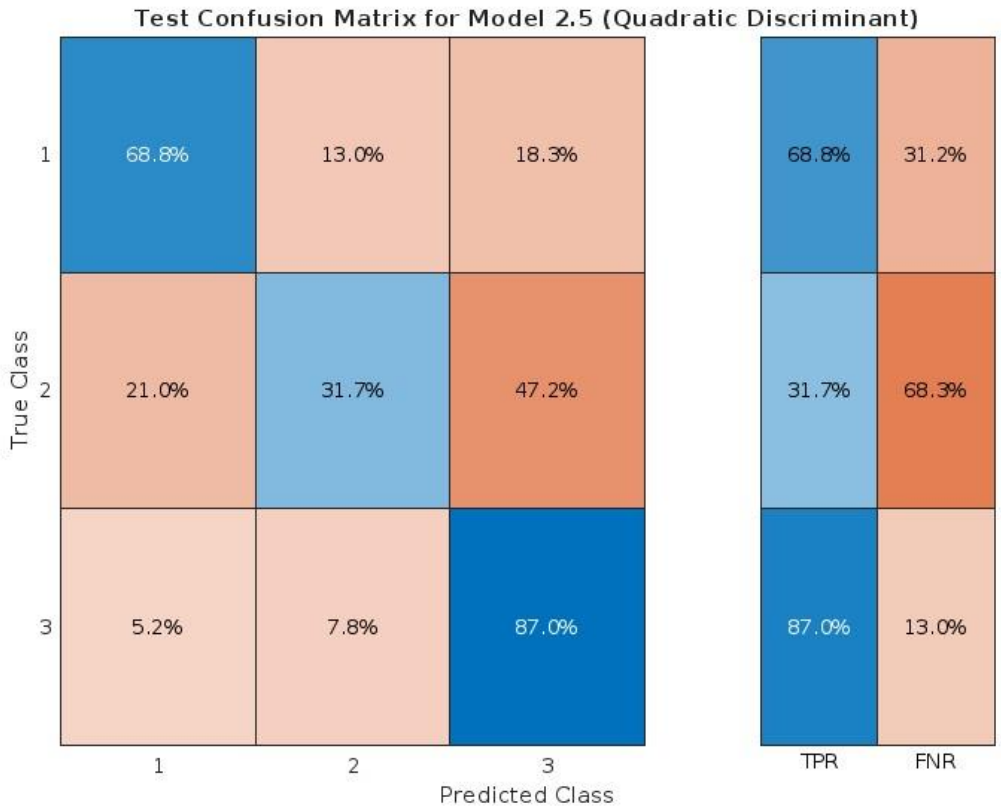


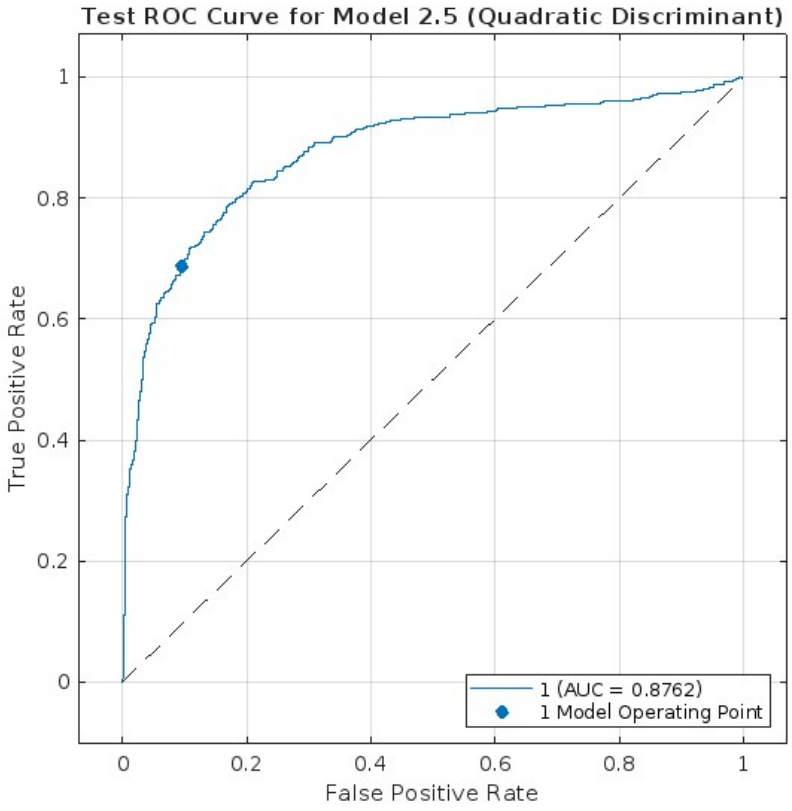
Quadratic Discriminant- Training graphs



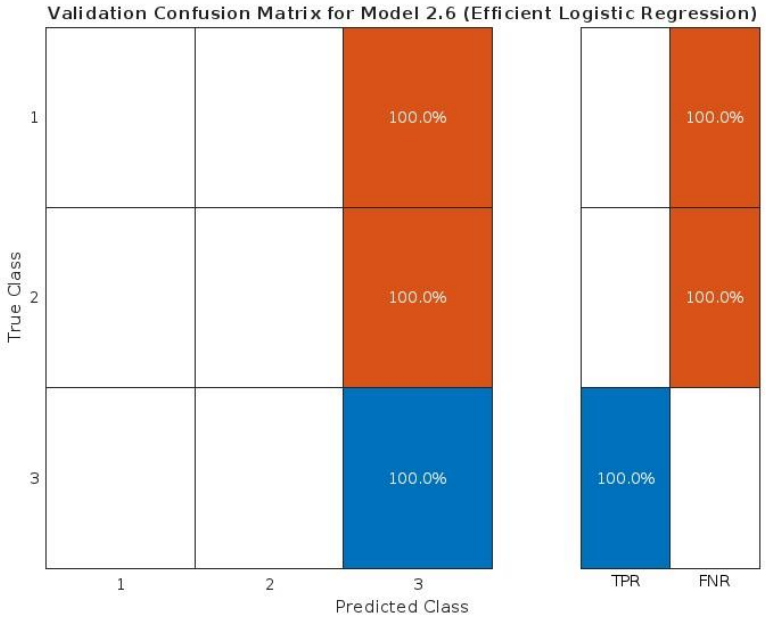


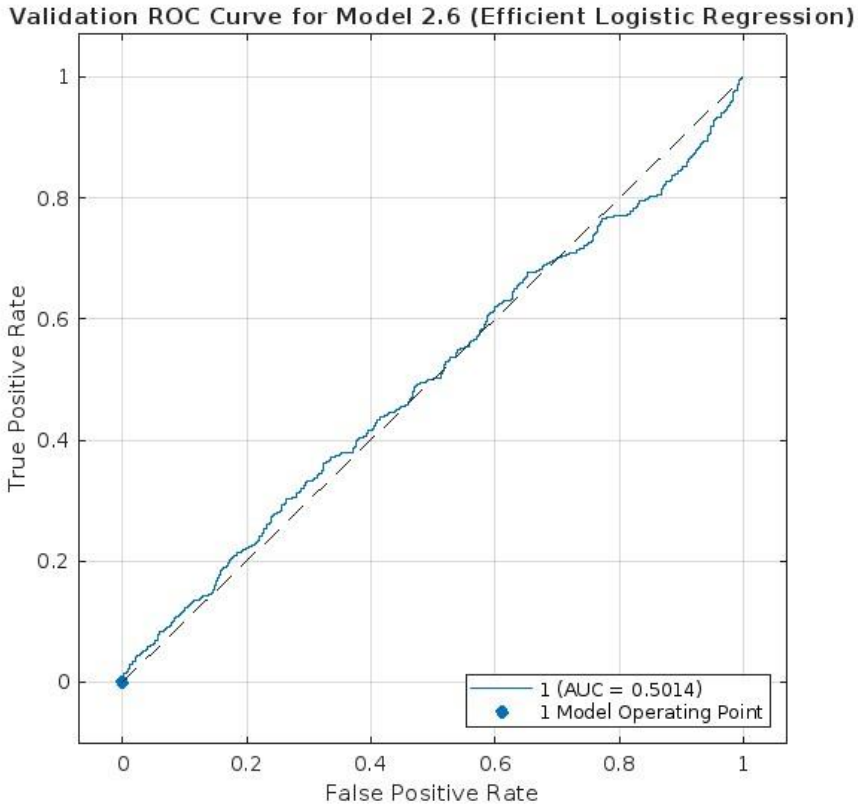
Testing graphs



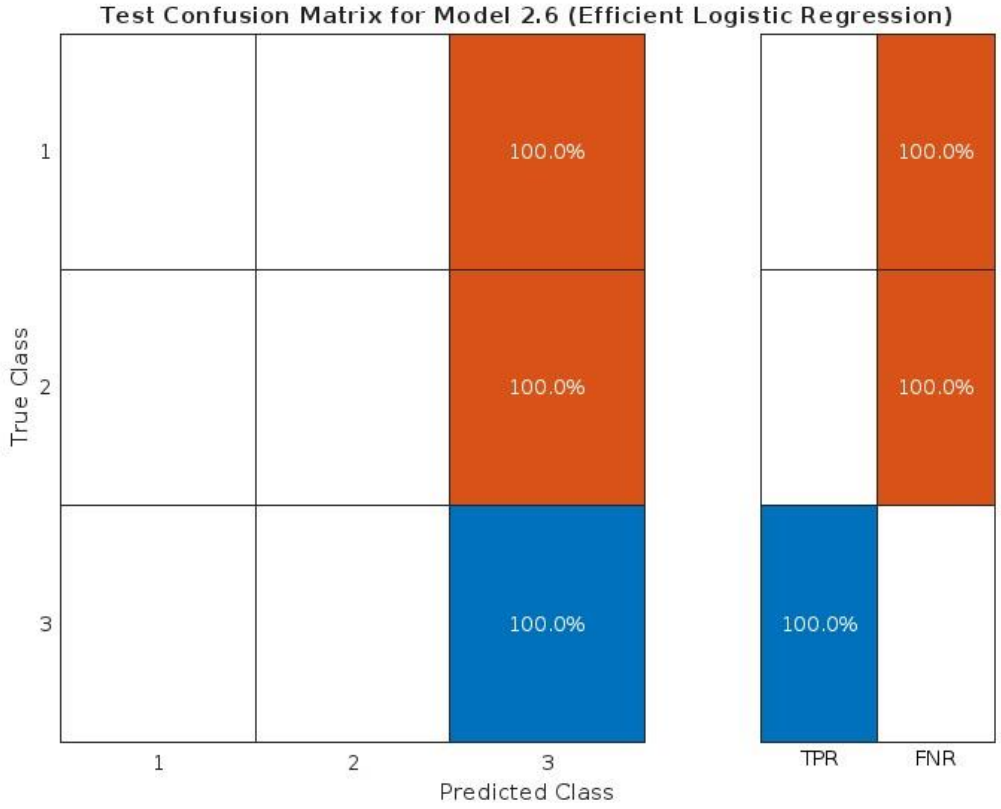


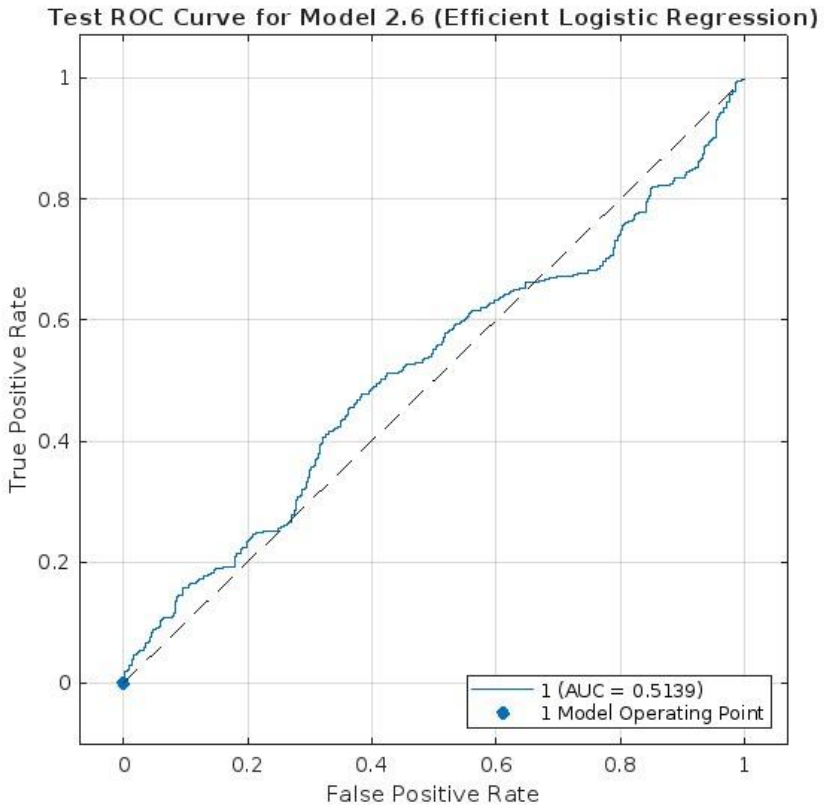
Efficient Logistic Regression- Training graphs



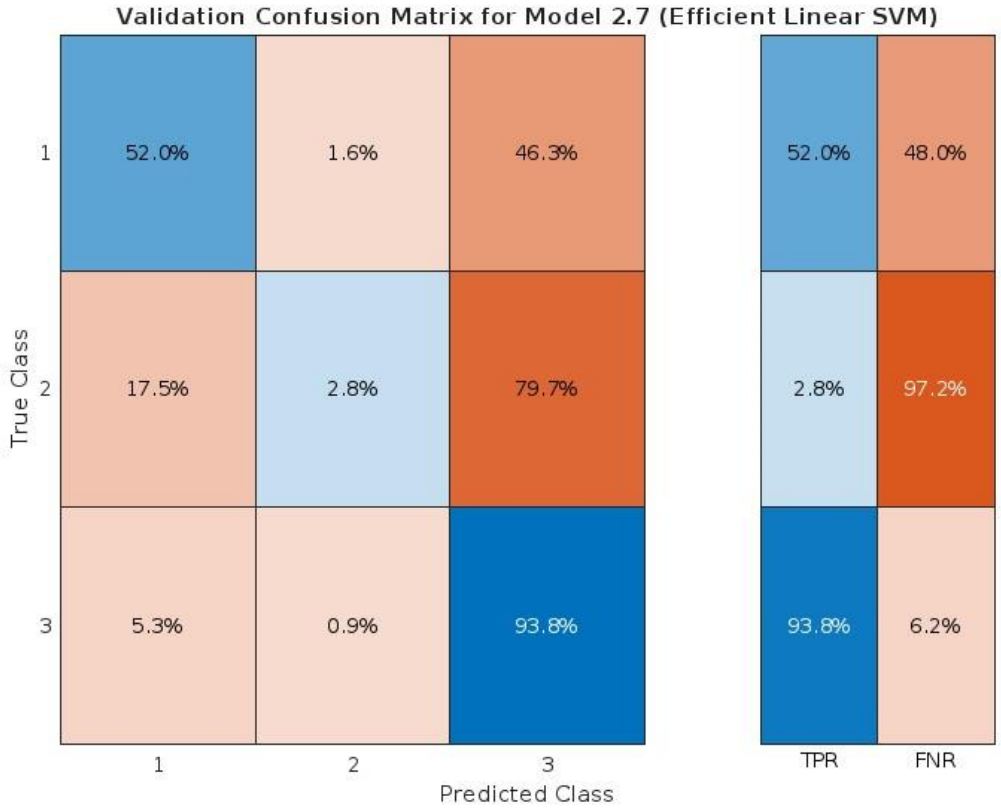


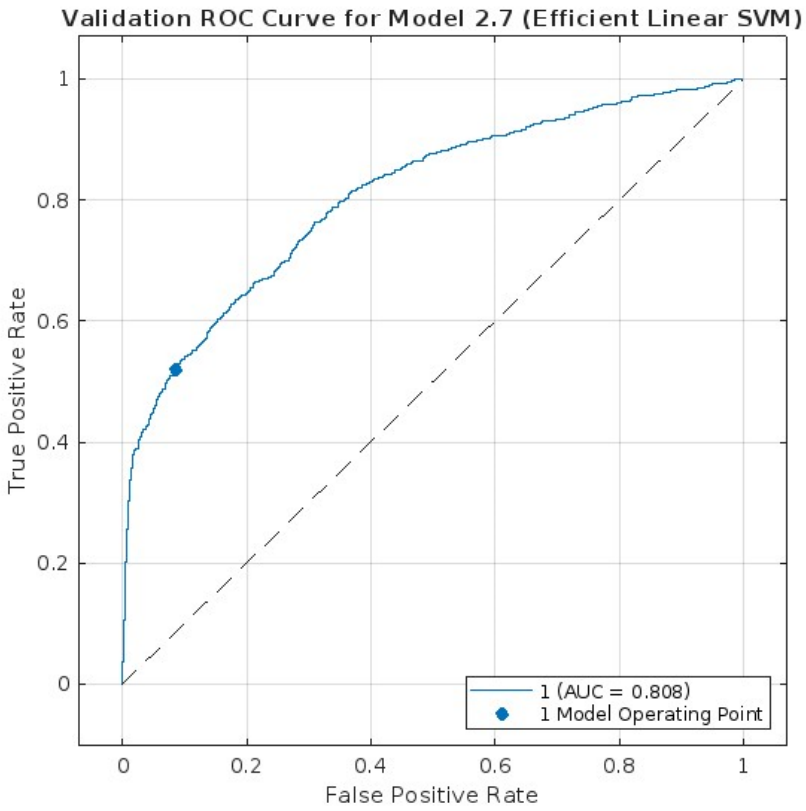
Testing graphs


Test Confusion Matrix for Model 2.6 (Efficient Logistic Regression)
True Class
Predicted Class
1
2
3
100.0%
100.0%
100.0%
100.0%
100.0%
100.0%
TPR
FNR

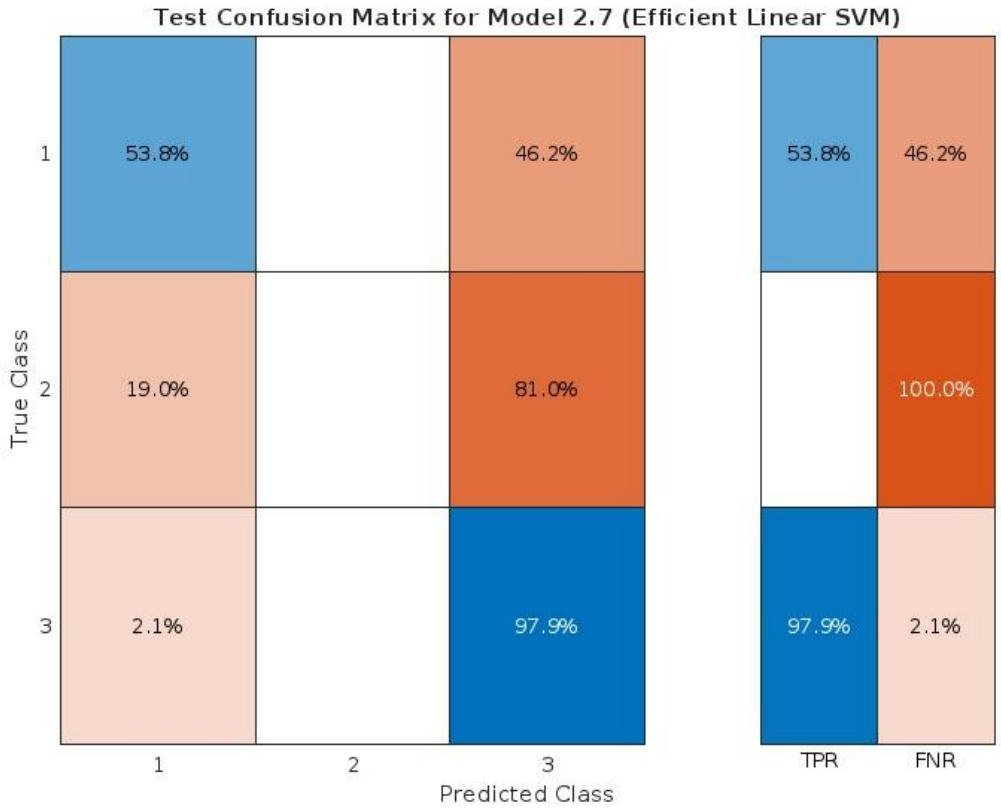

Test ROC Curve for Model 2.6 (Efficient Logistic Regression)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1 (AUC = 0.5139)
1 Model Operating Point

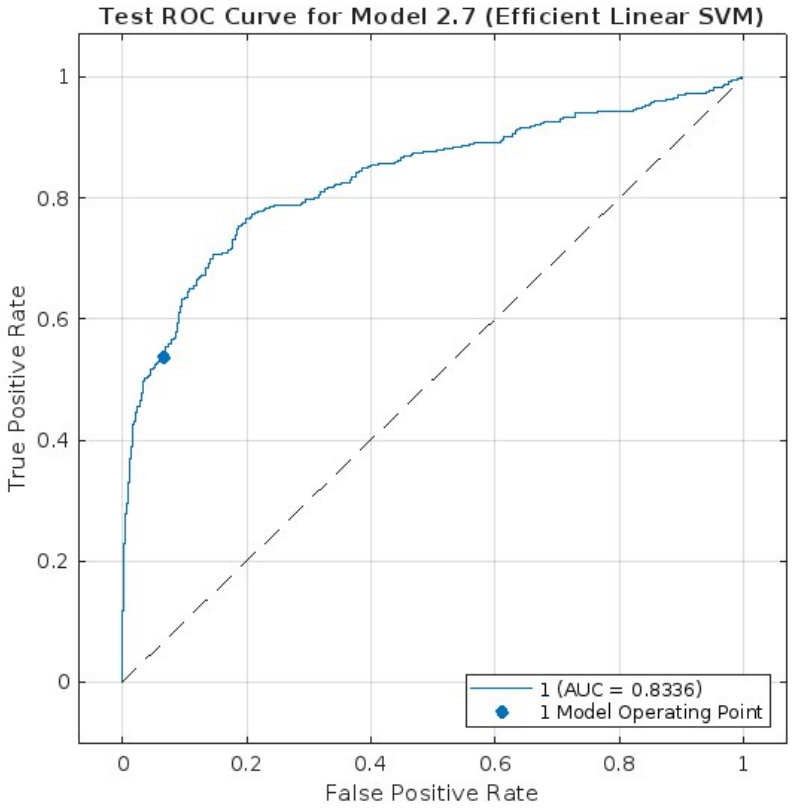
Efficient Linear SVM - Training Graphs



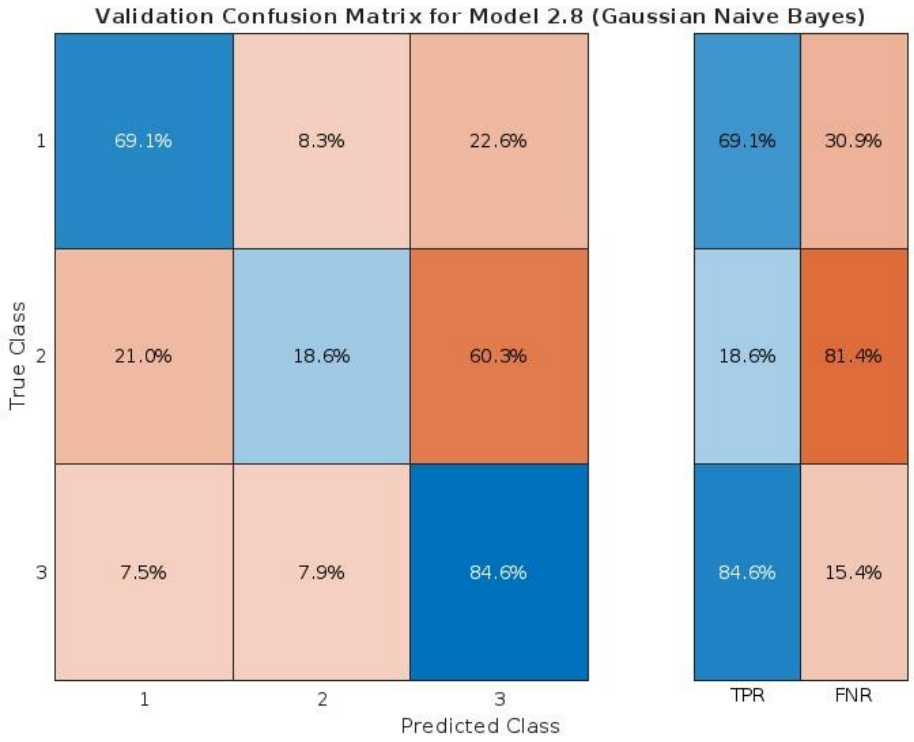


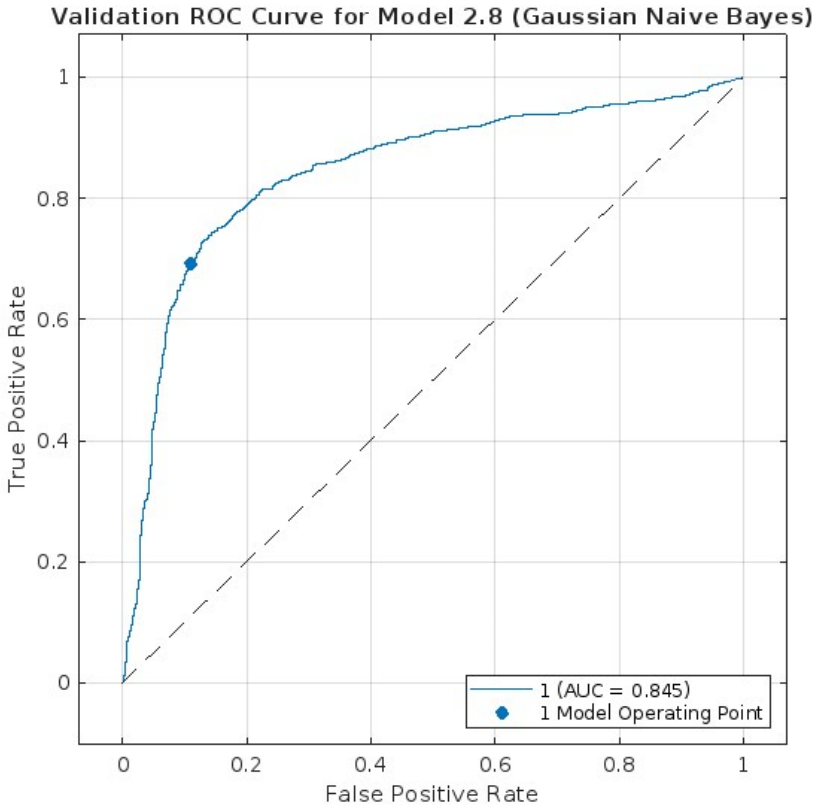
Testing graphs



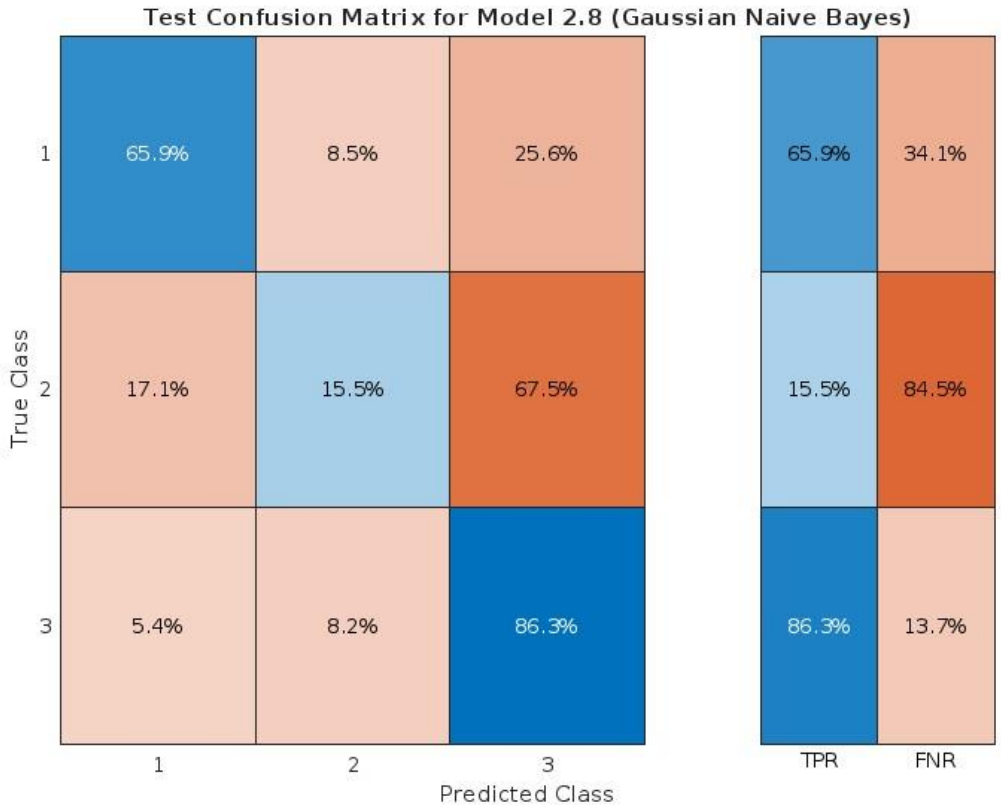


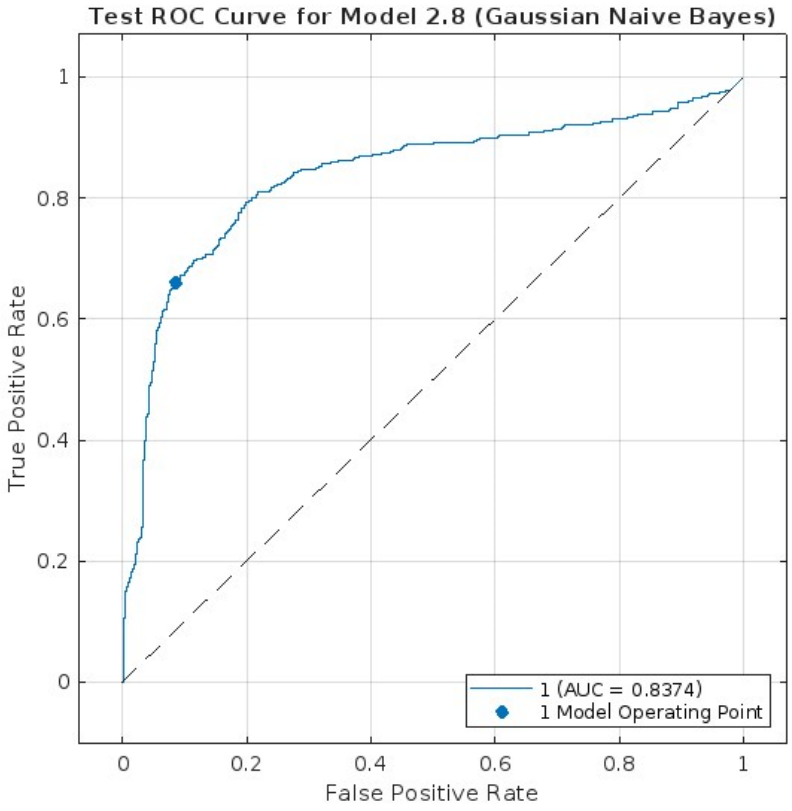
Gaussian Naive Bayes – Training graph



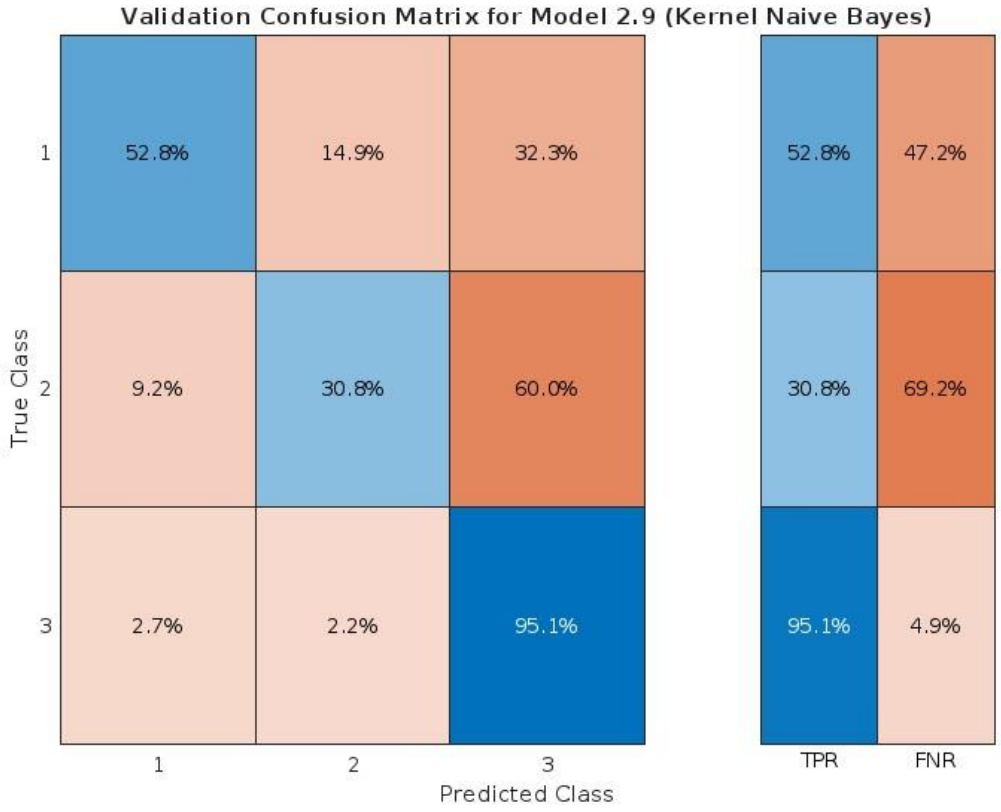


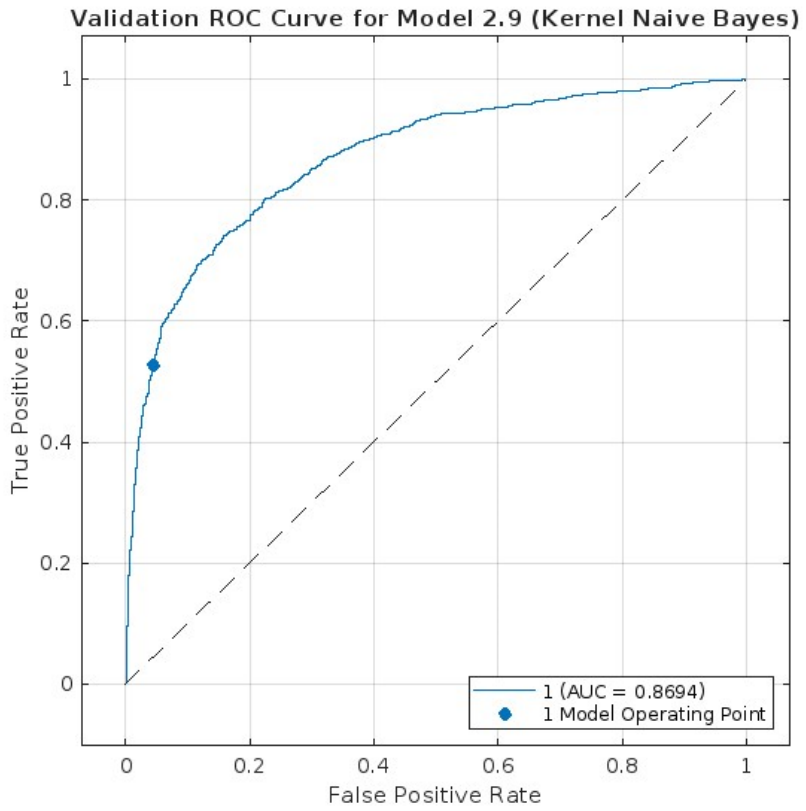
Testing graphs



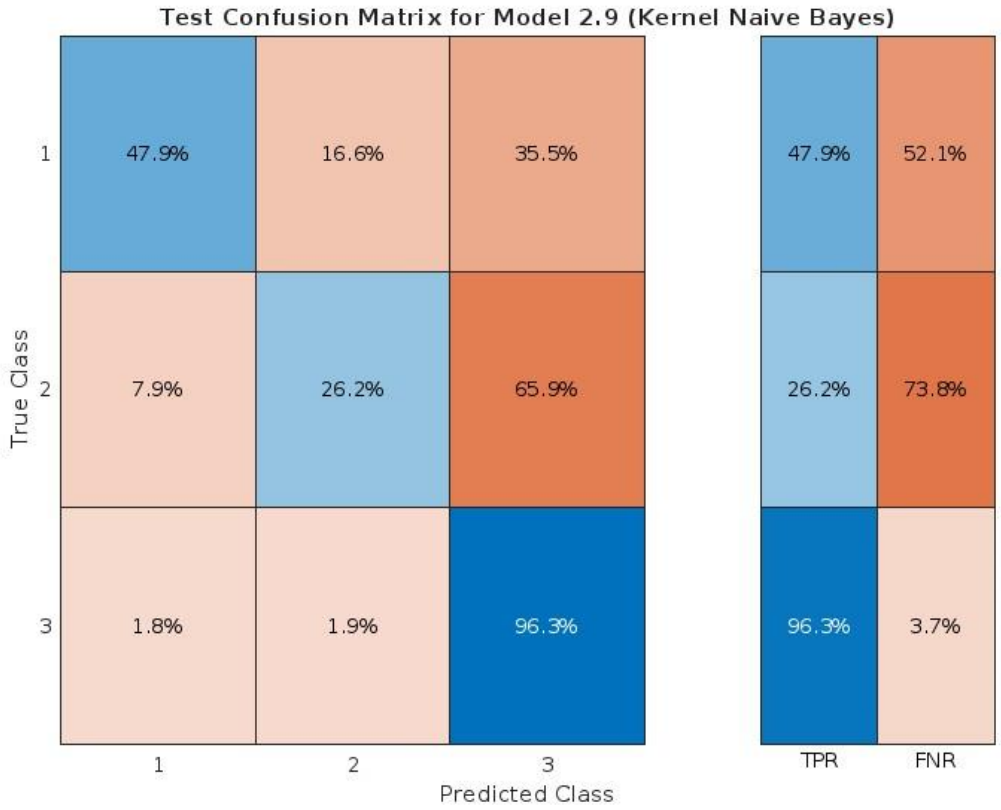


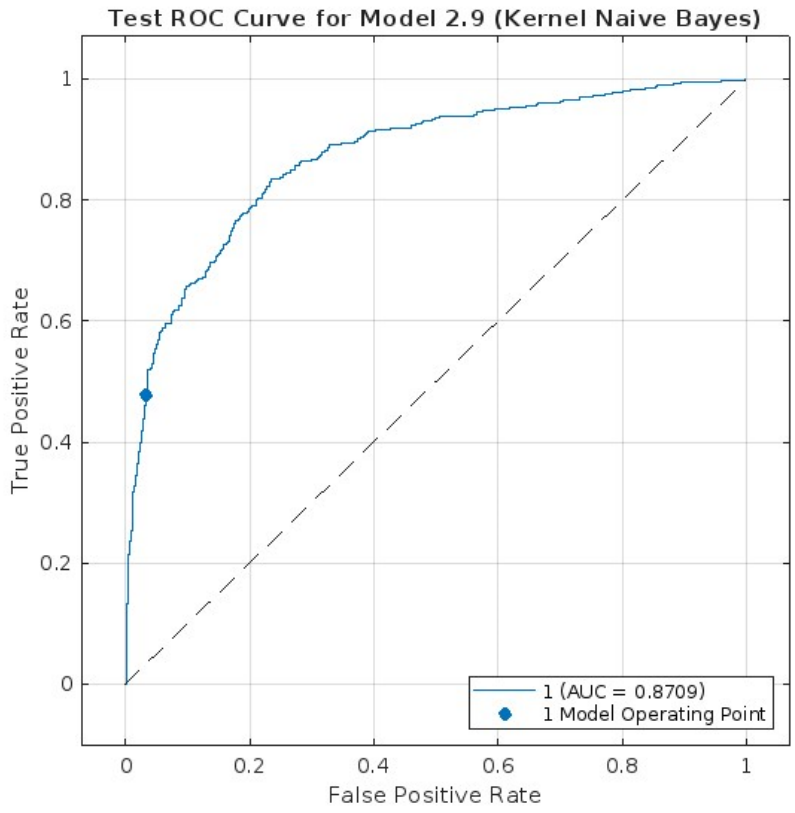
Kernel Naive Bayes - Training graph



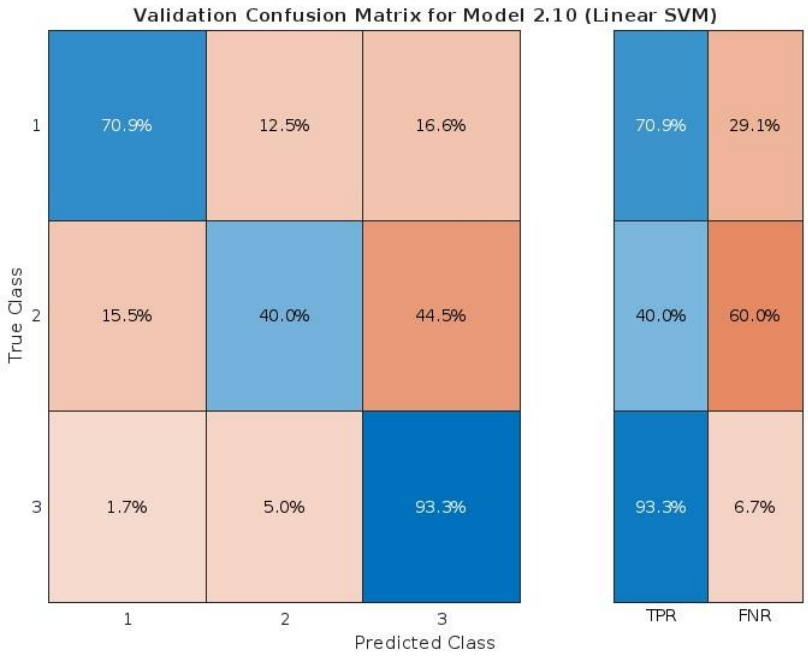


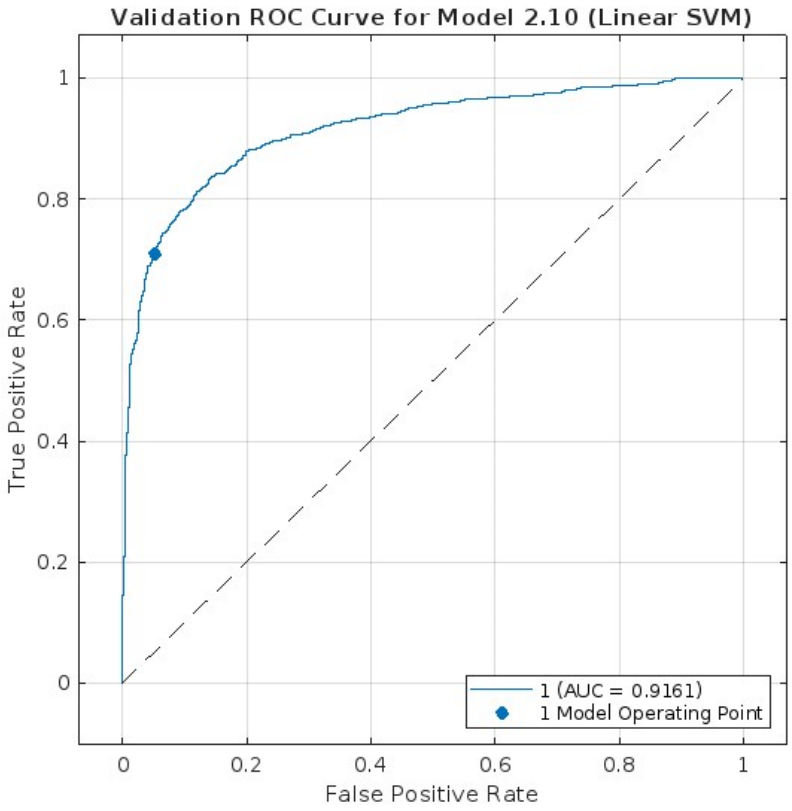
Testing graphs



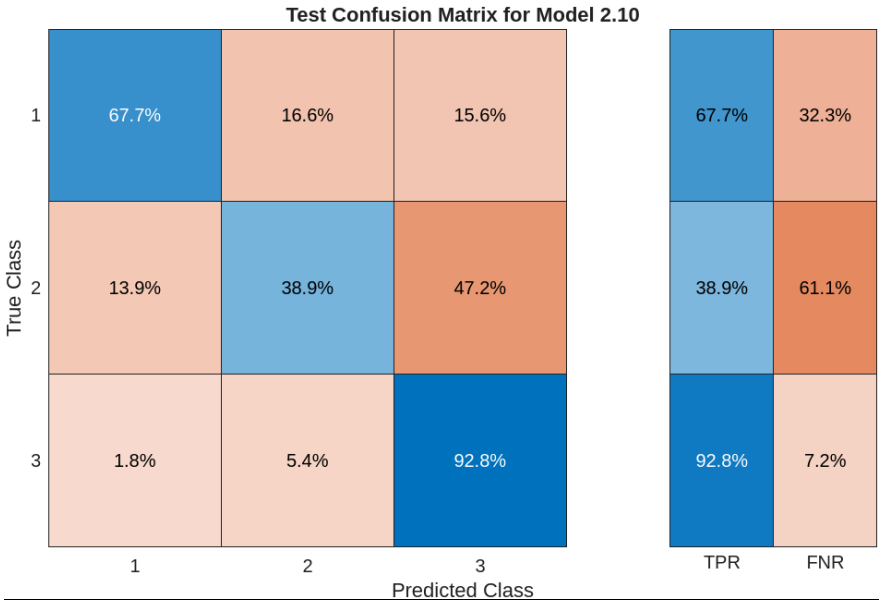


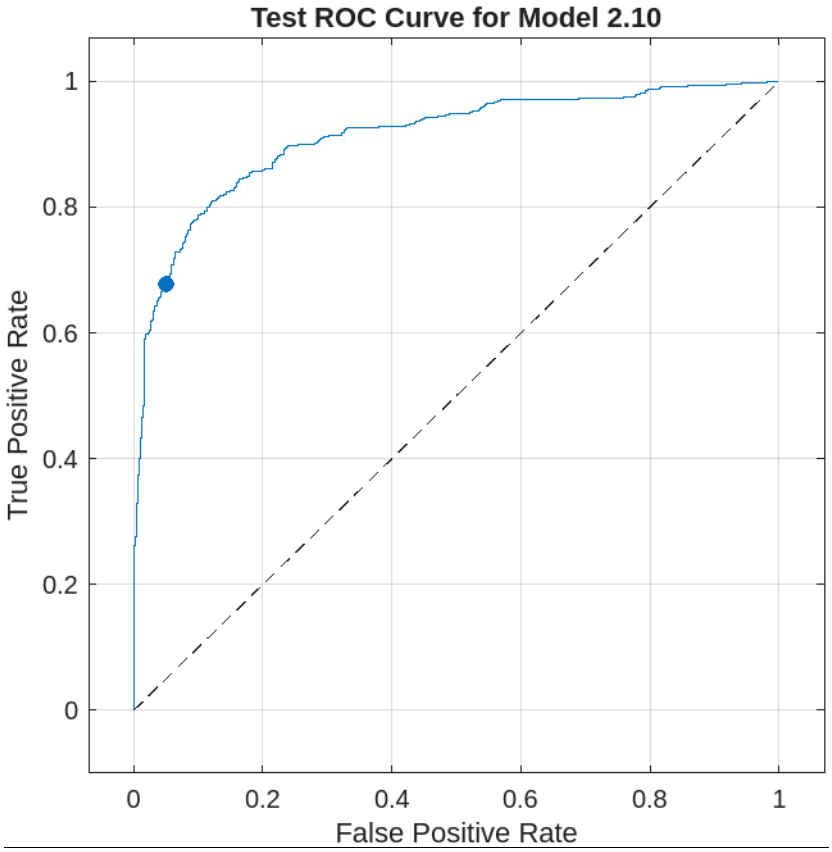
Linear SVM


Validation Confusion Matrix for Model 2.10 (Linear SVM)
70.9%
12.5%
16.6%
15.5%
40.0%
44.5%
1.7%
5.0%
93.3%
70.9%
29.1%
40.0%
60.0%
93.3%
6.7%
True Class
Predicted Class
TPR
FNR

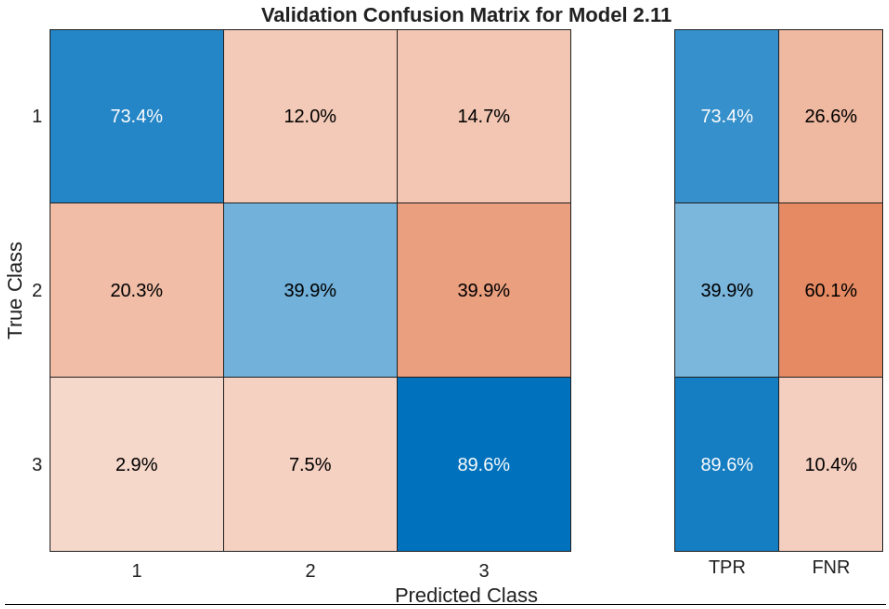

Validation ROC Curve for Model 2.10 (Linear SVM)
True Positive Rate
False Positive Rate
1 (AUC = 0.9161)
1 Model Operating Point

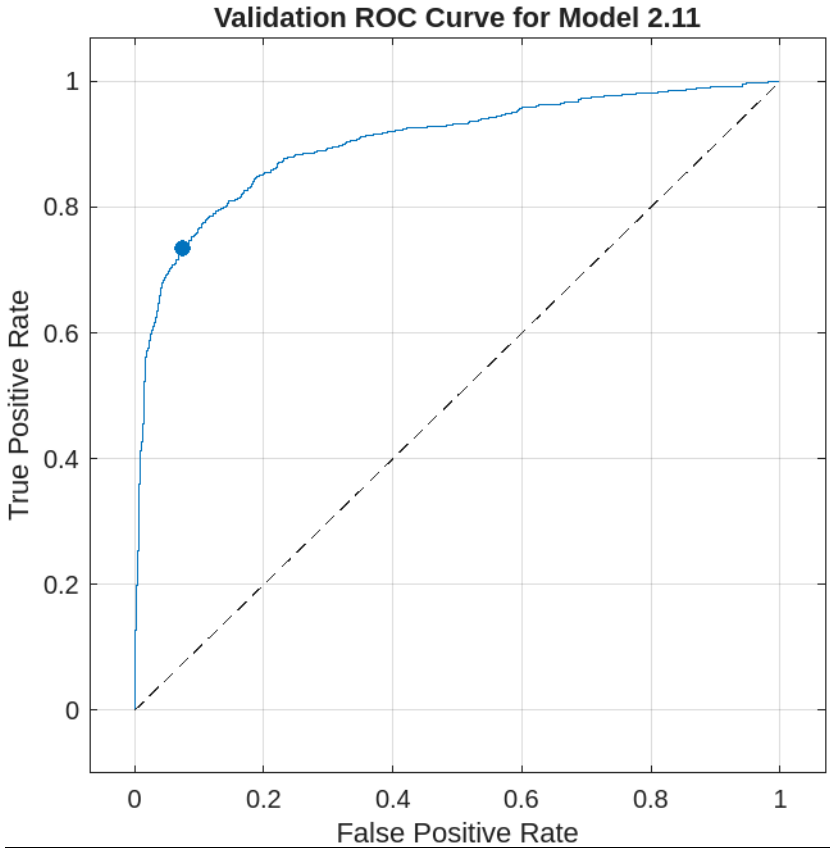
Testing graphs



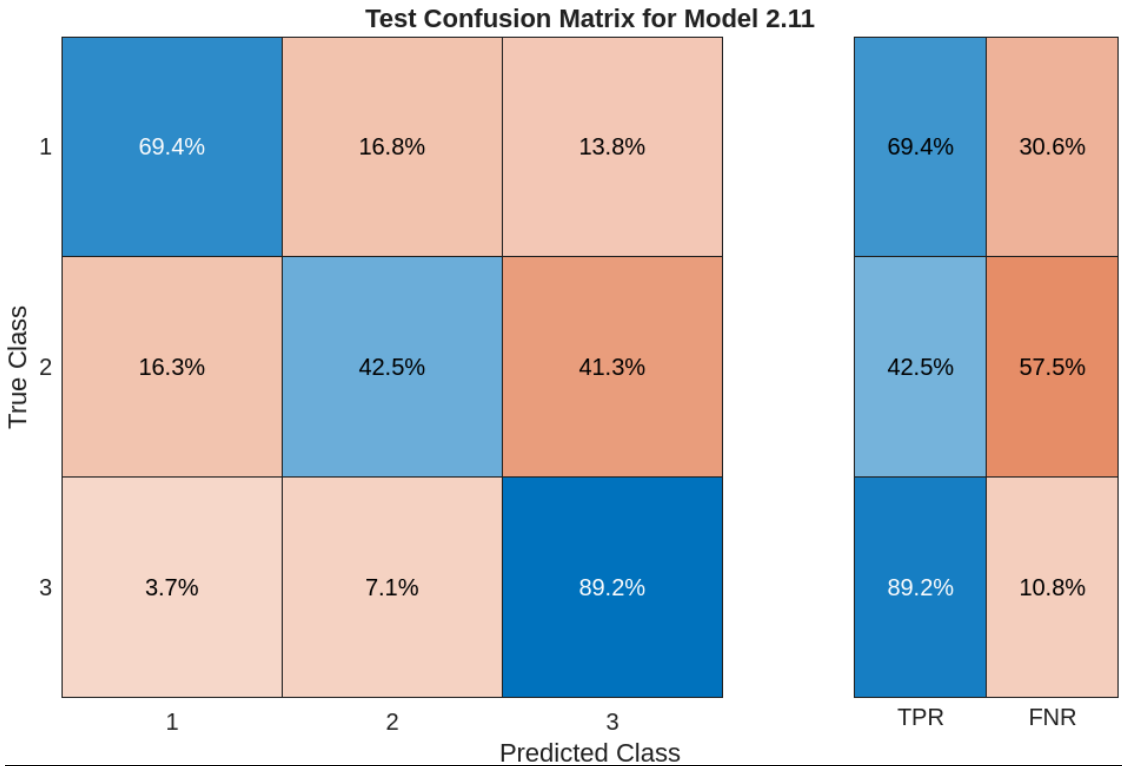


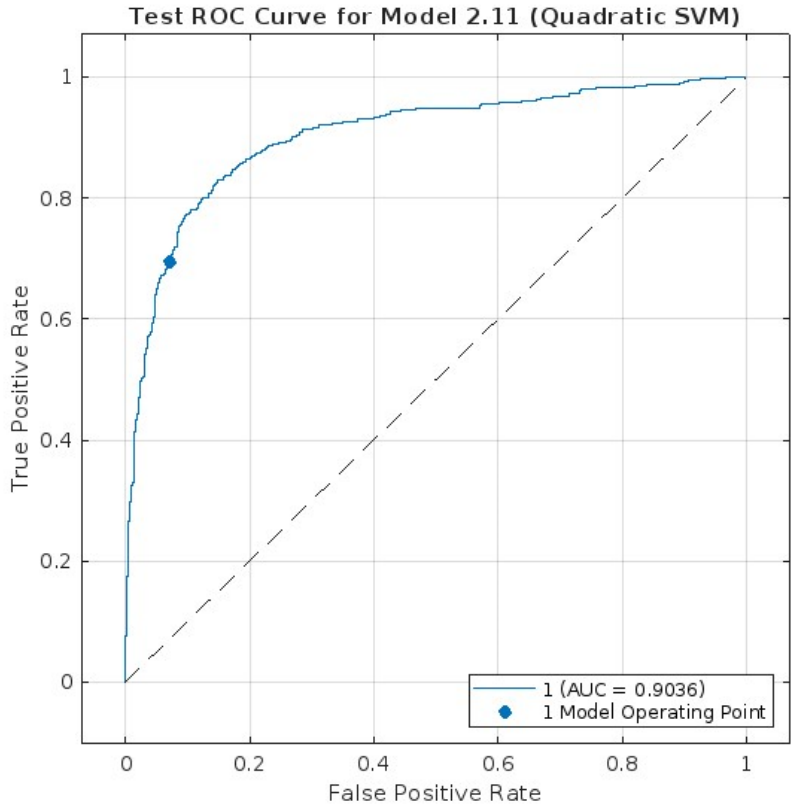
Quadratic SVM



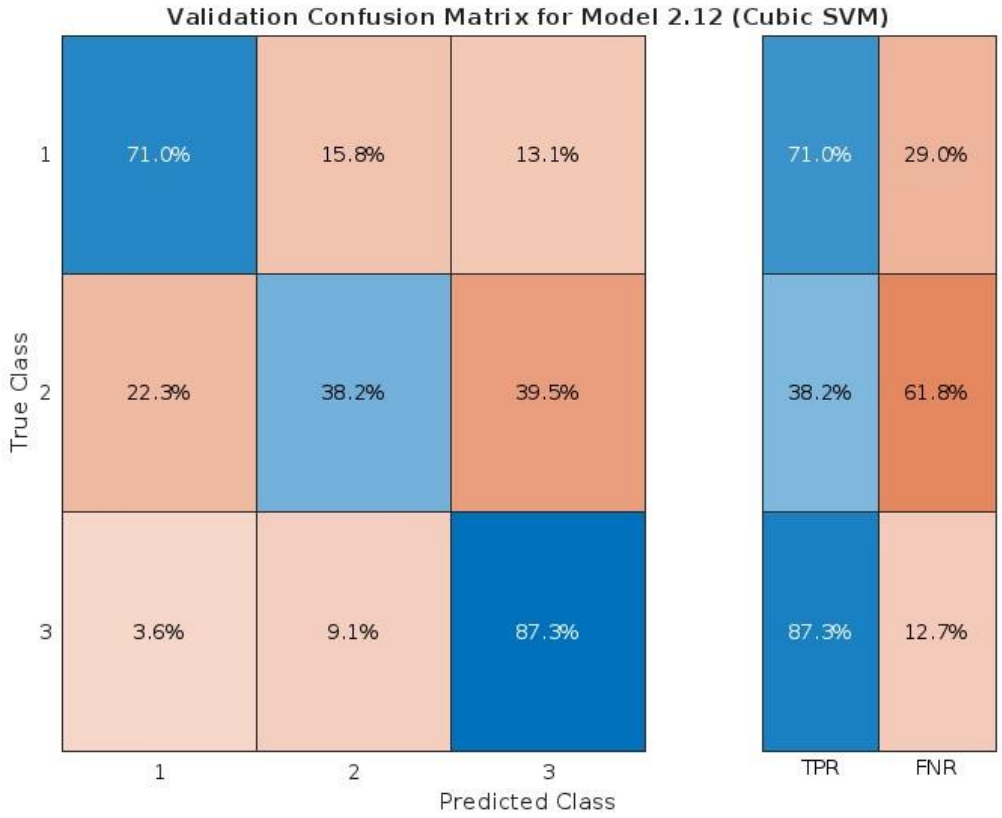


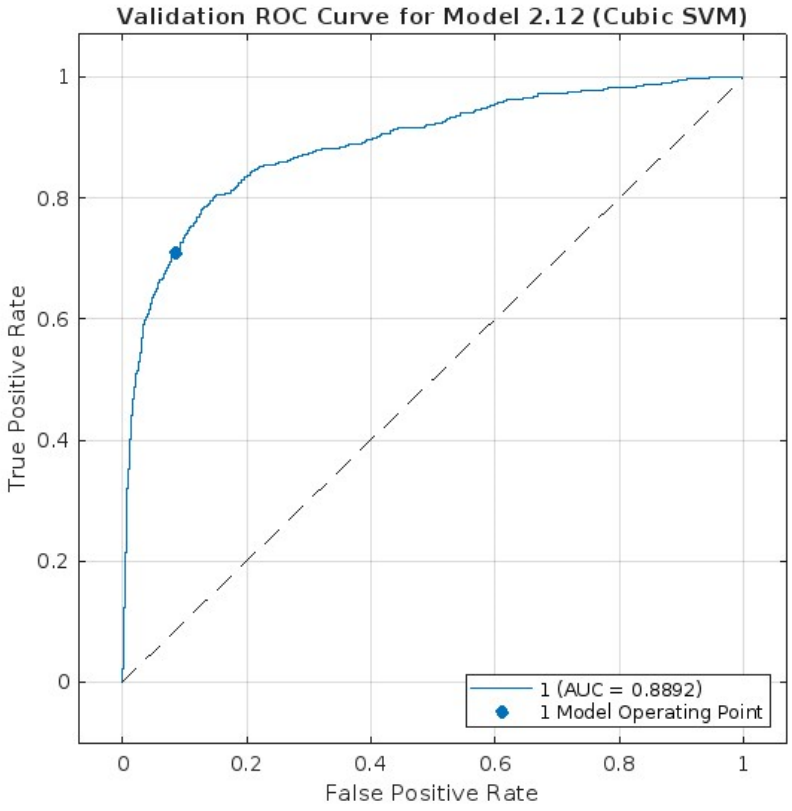
Testing graphs


Test Confusion Matrix for Model 2.11
True Class
1
2
3
69.4%
16.8%
13.8%
16.3%
42.5%
41.3%
3.7%
7.1%
89.2%
1
2
3
Predicted Class
69.4%
30.6%
42.5%
57.5%
89.2%
10.8%
TPR
FNR

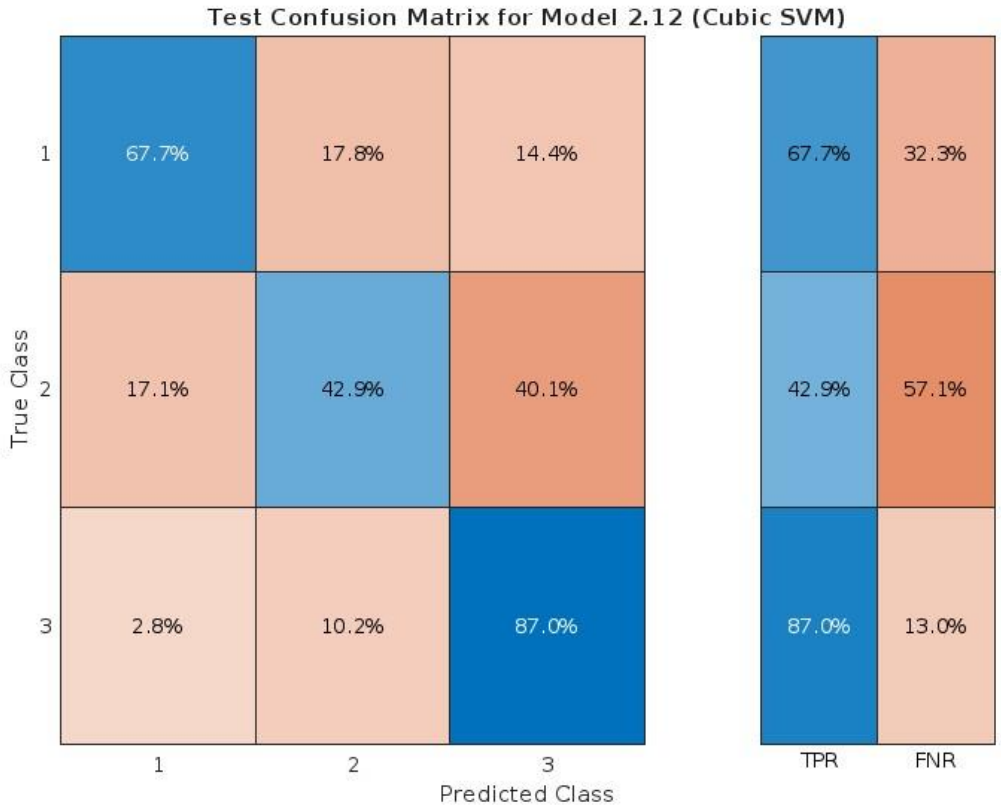

Test ROC Curve for Model 2.11 (Quadratic SVM)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1 (AUC = 0.9036)
1 Model Operating Point

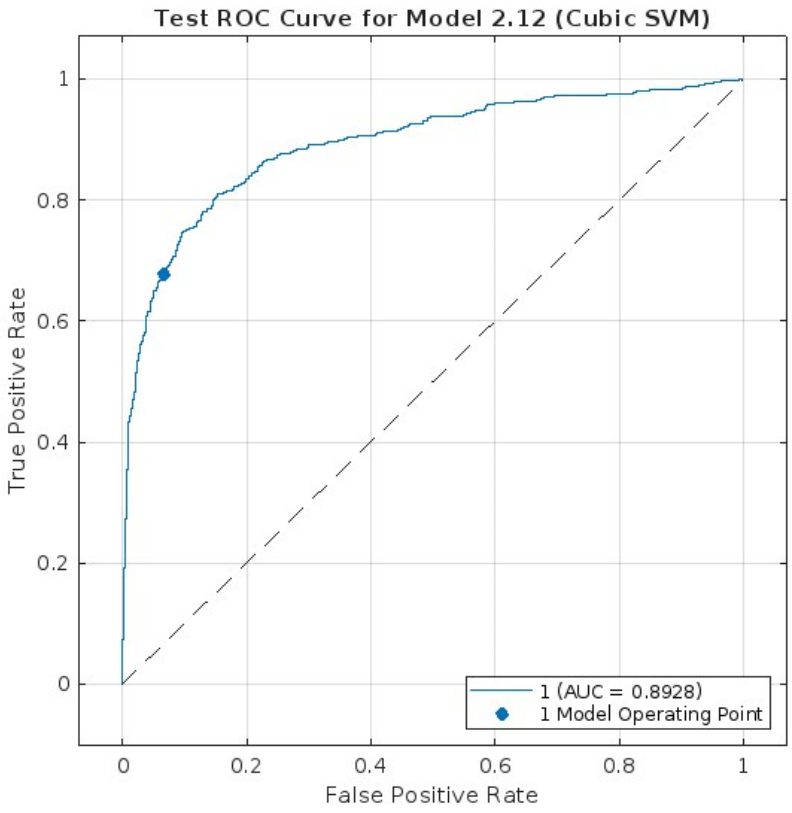
Cubic SVM- Training graph



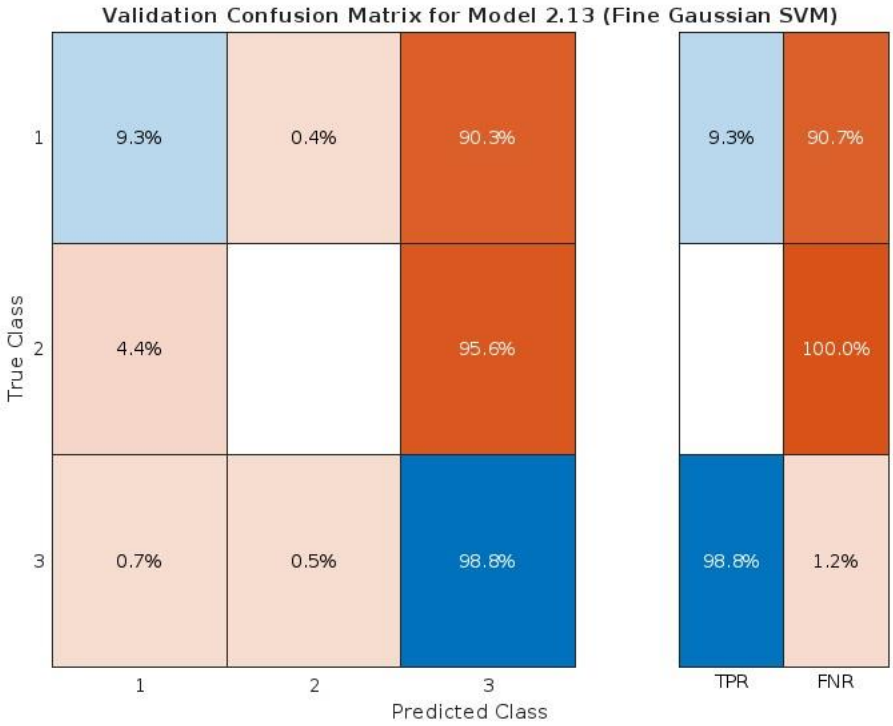


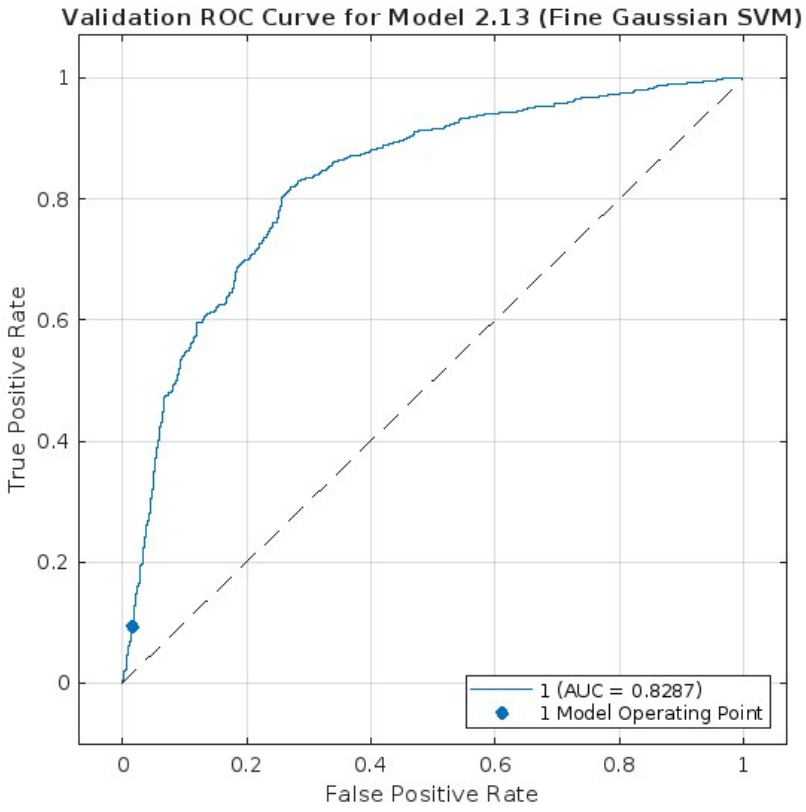
Testing graph



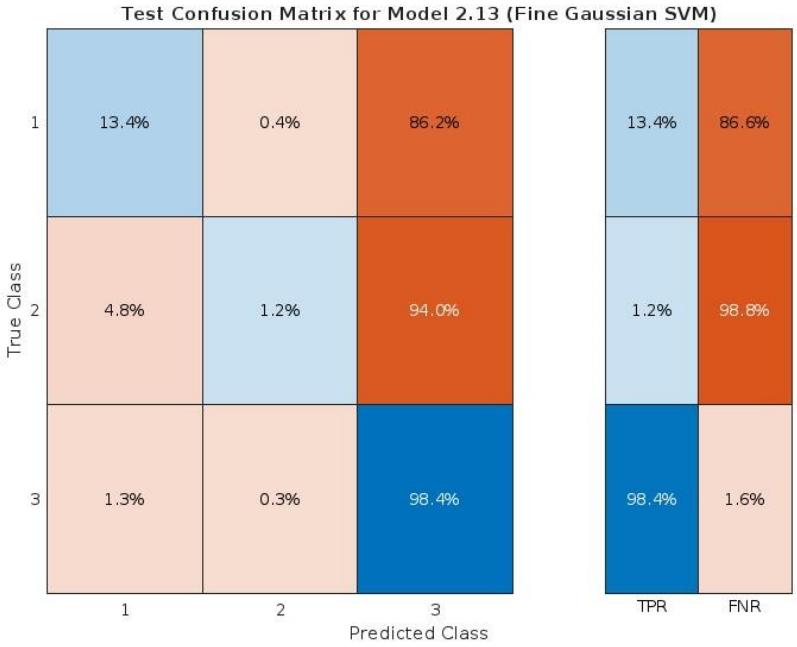


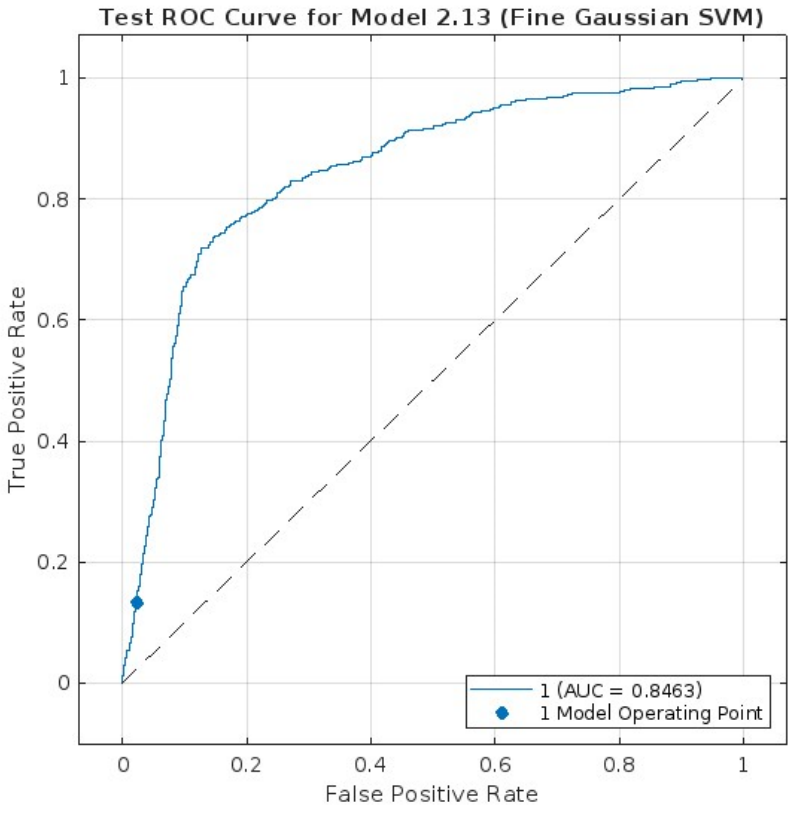
Fine Gaussian SVM-Training graph



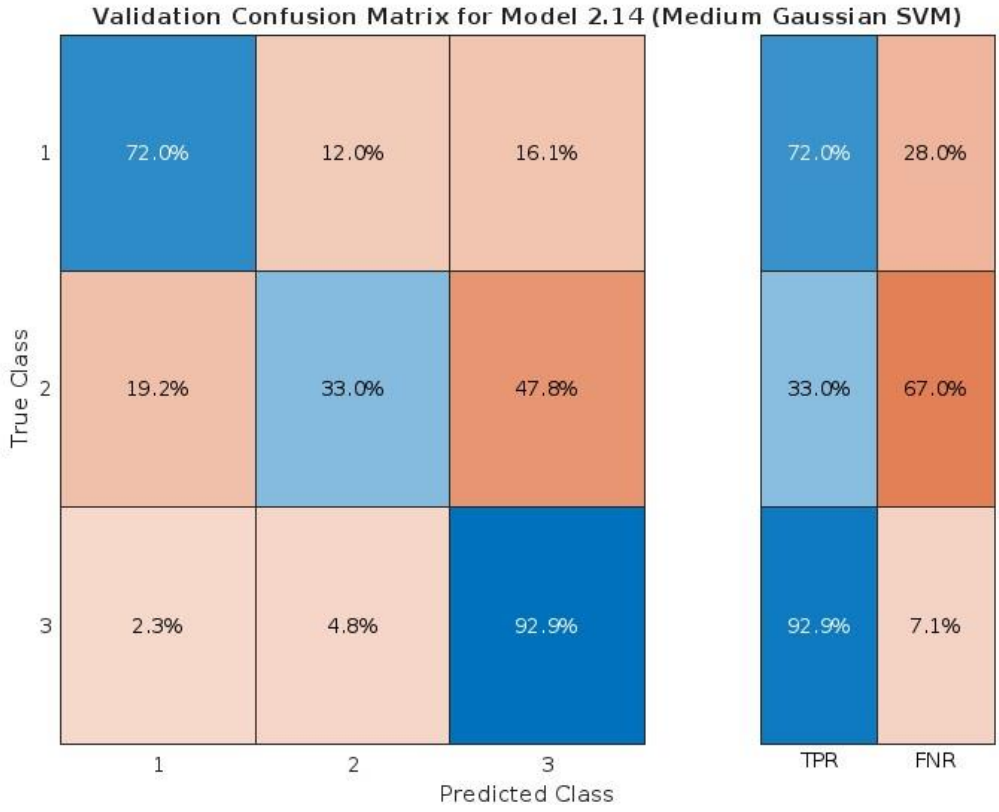


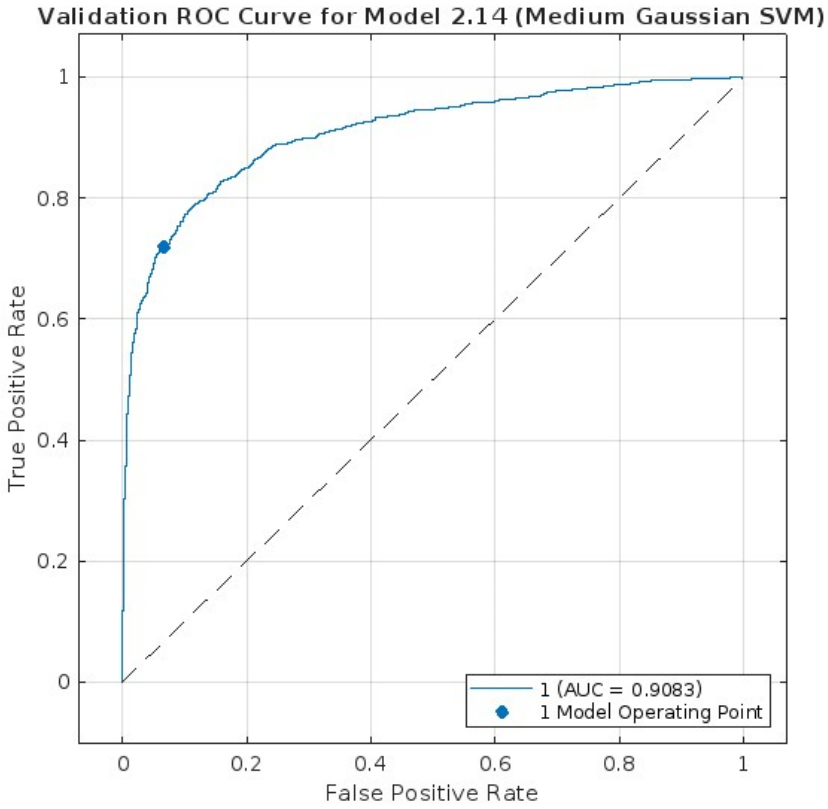
Testing graph



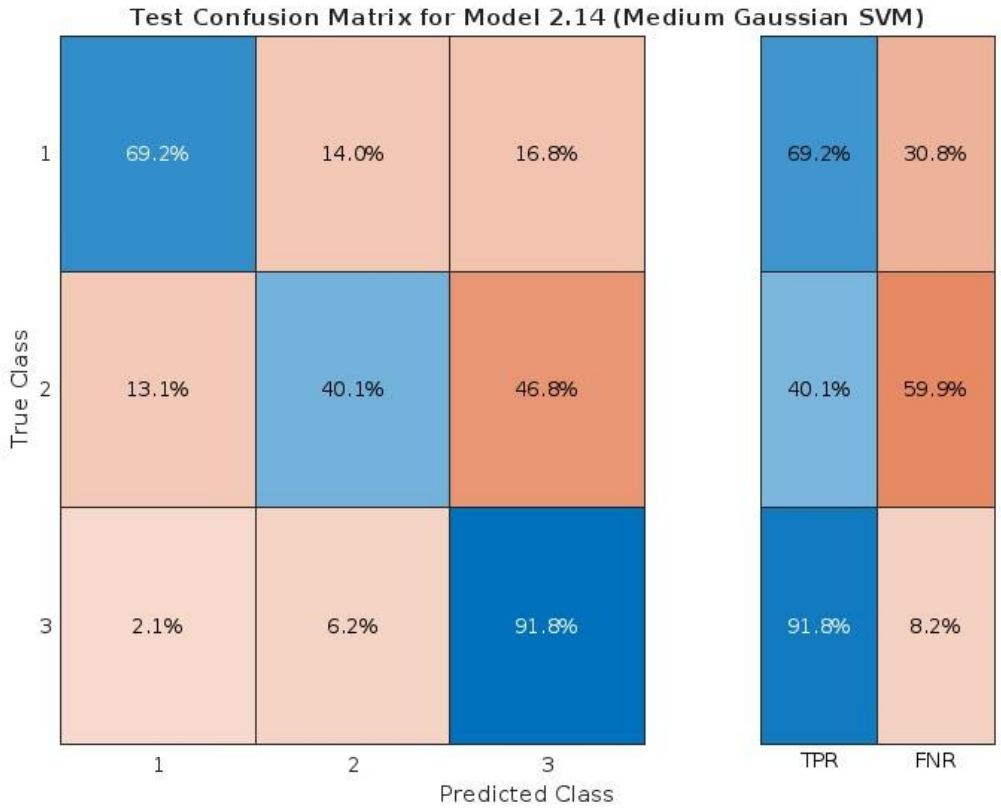


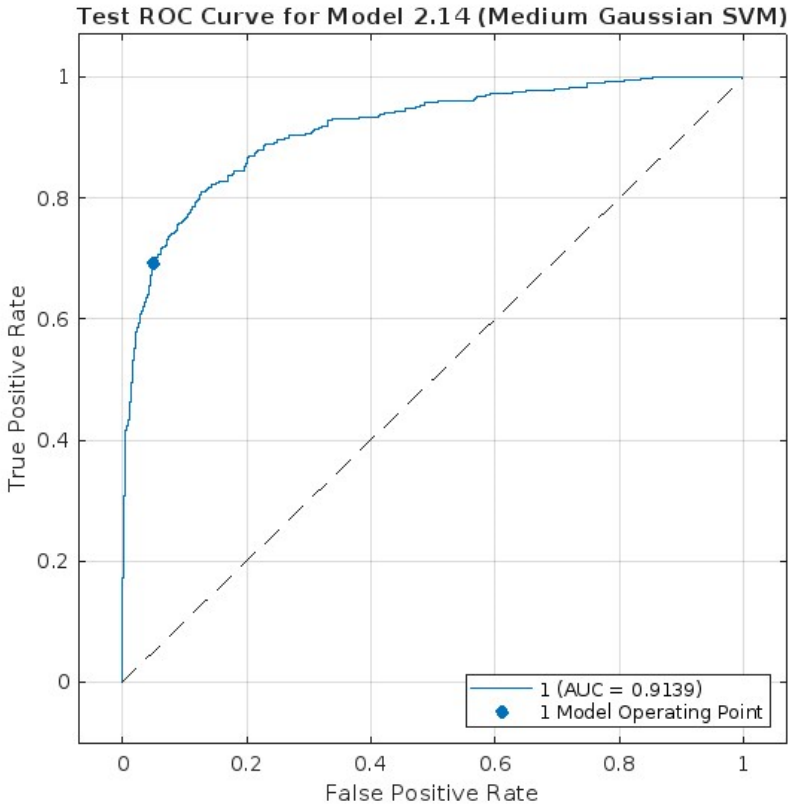
Medium Gaussian SVM



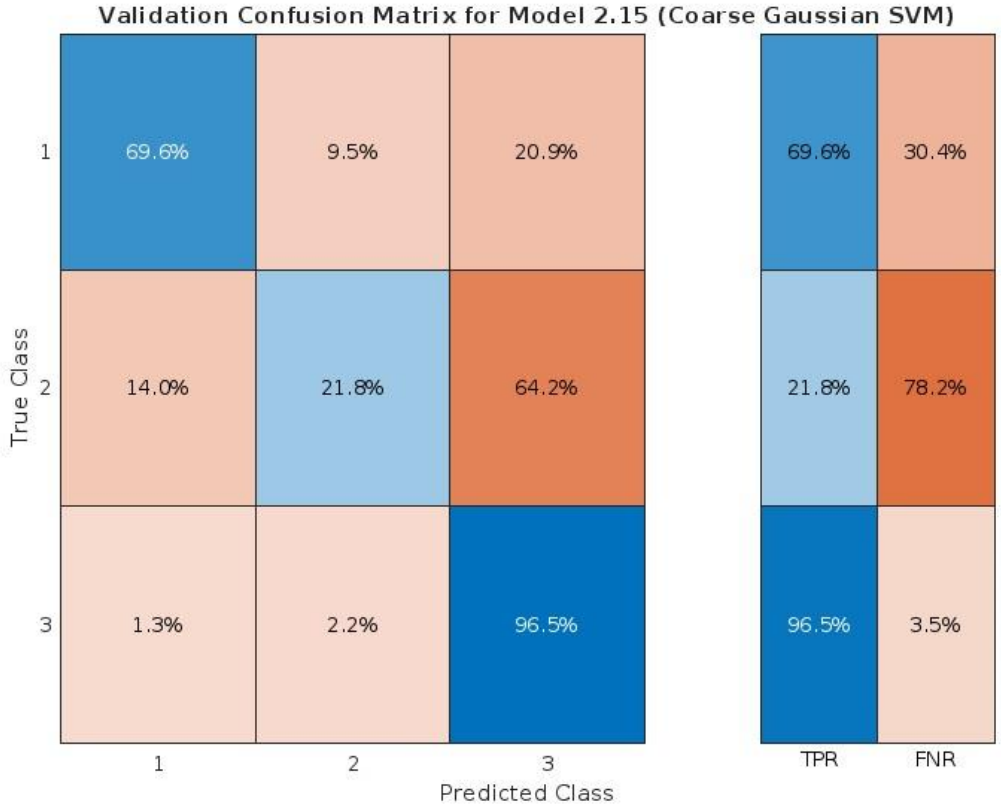


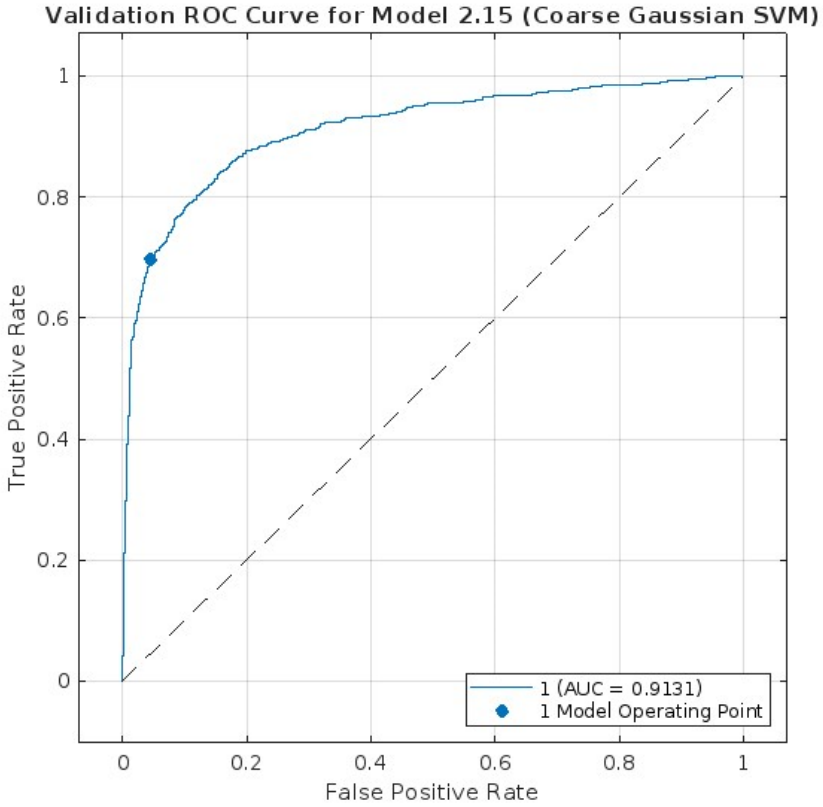
Testing graph



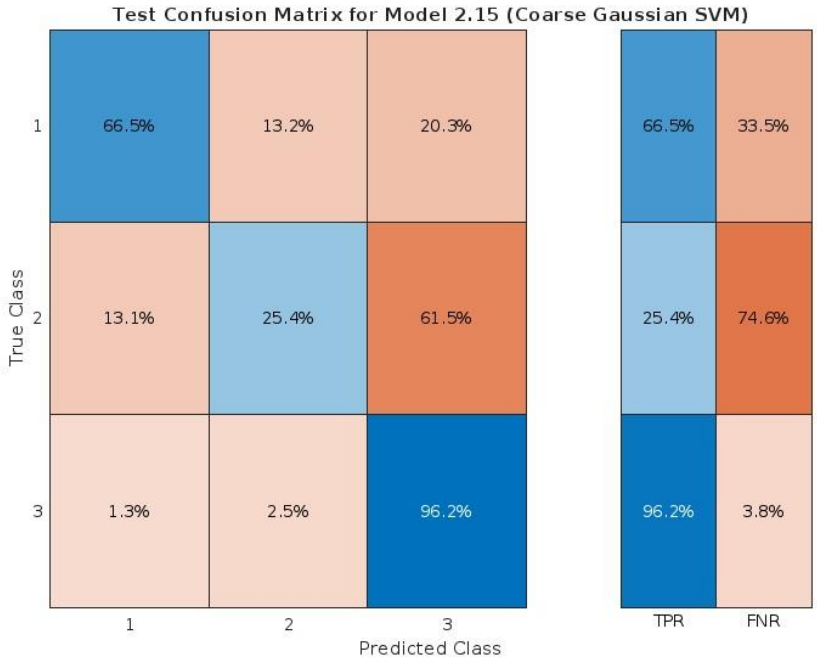


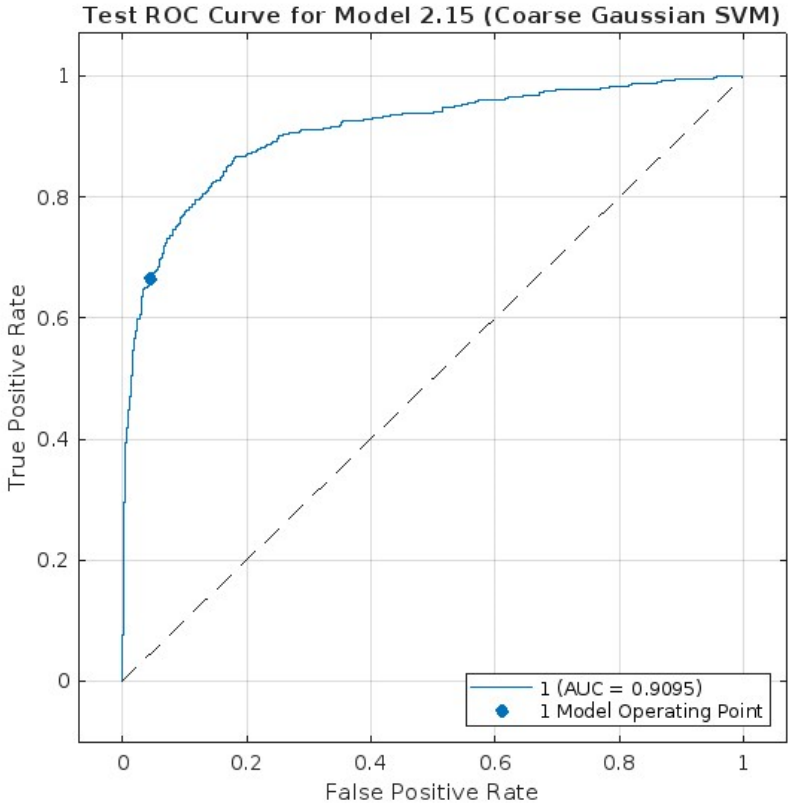
Coarse Gaussian SVM-Training graph



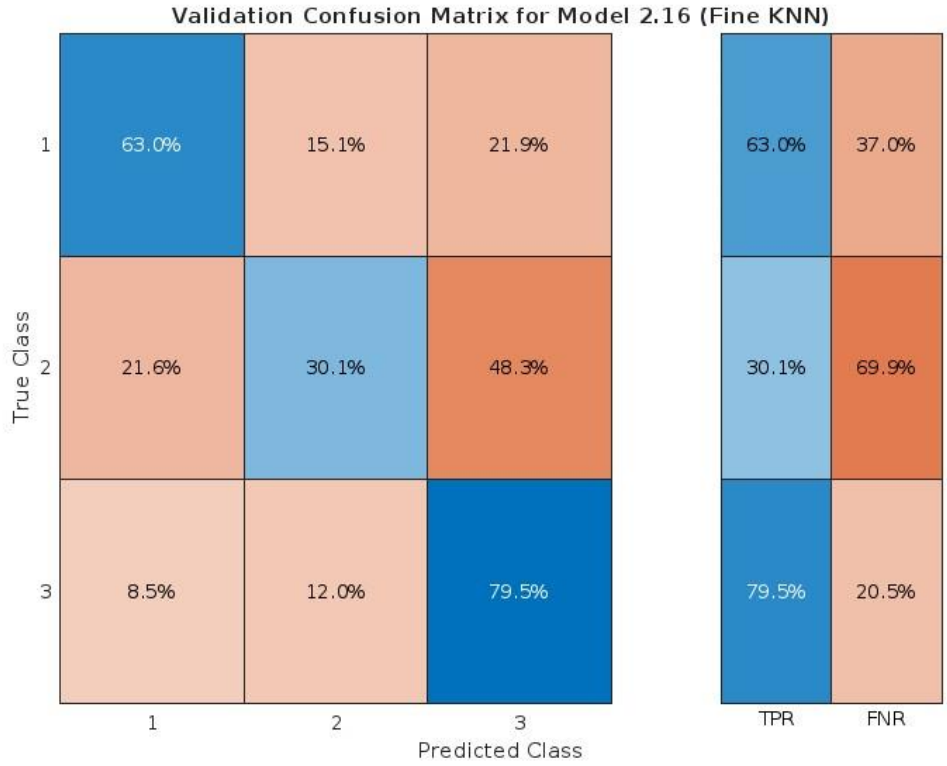


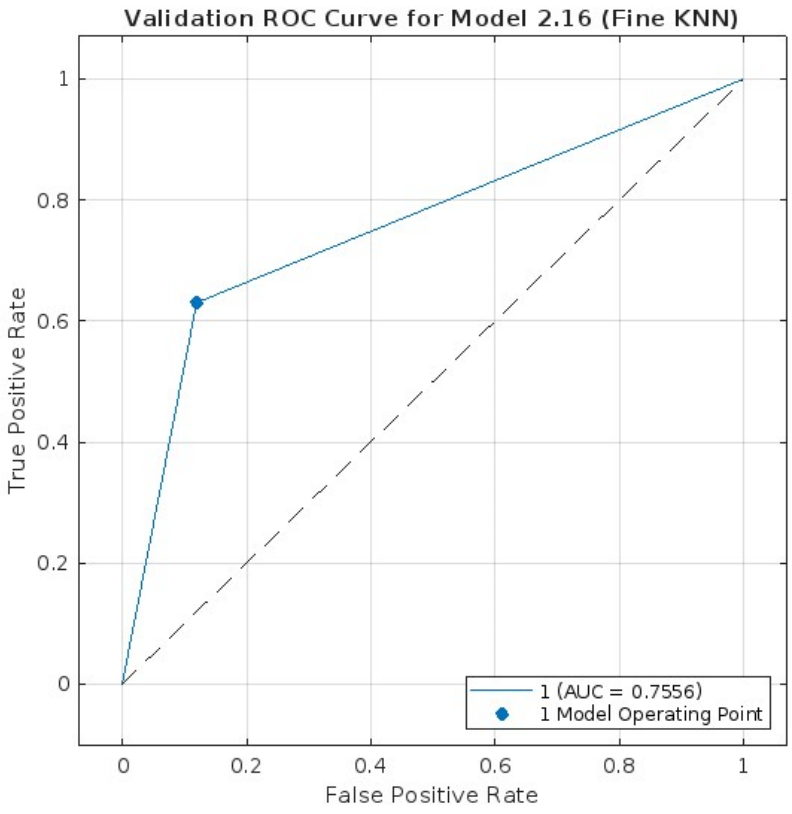
Testing graph



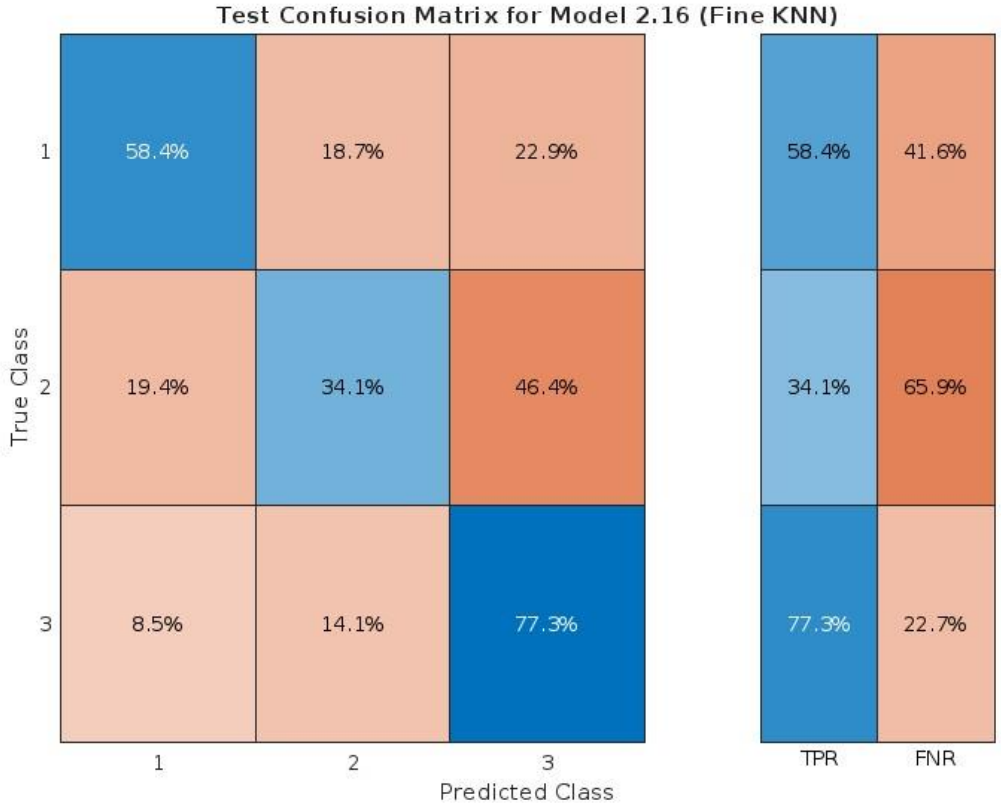

Test ROC Curve for Model 2.15 (Coarse Gaussian SVM)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1 (AUC = 0.9095)
1 Model Operating Point

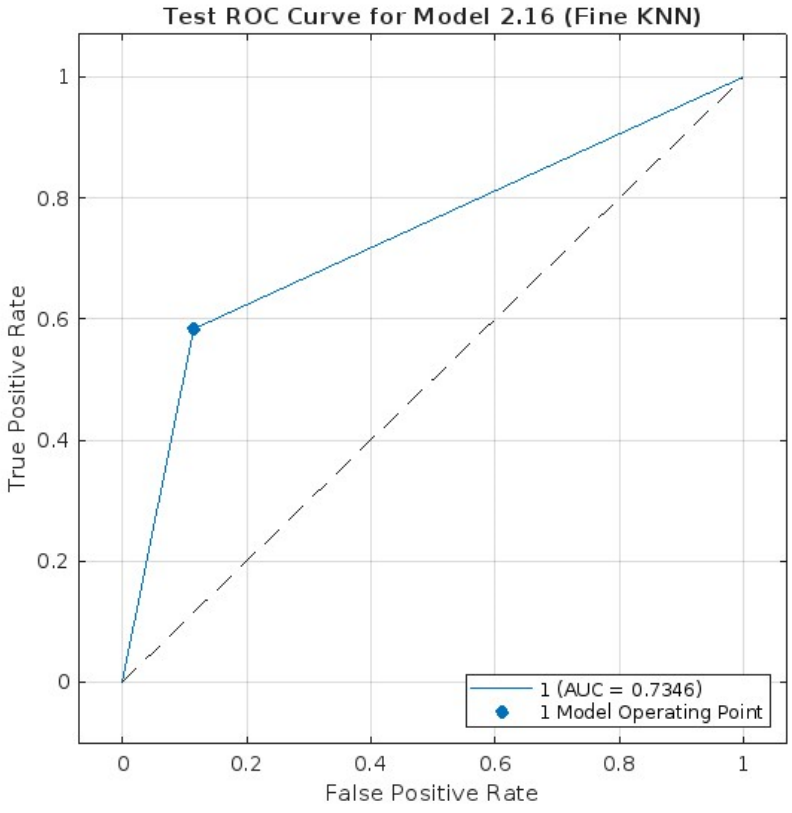
Fine KNN-Training graph



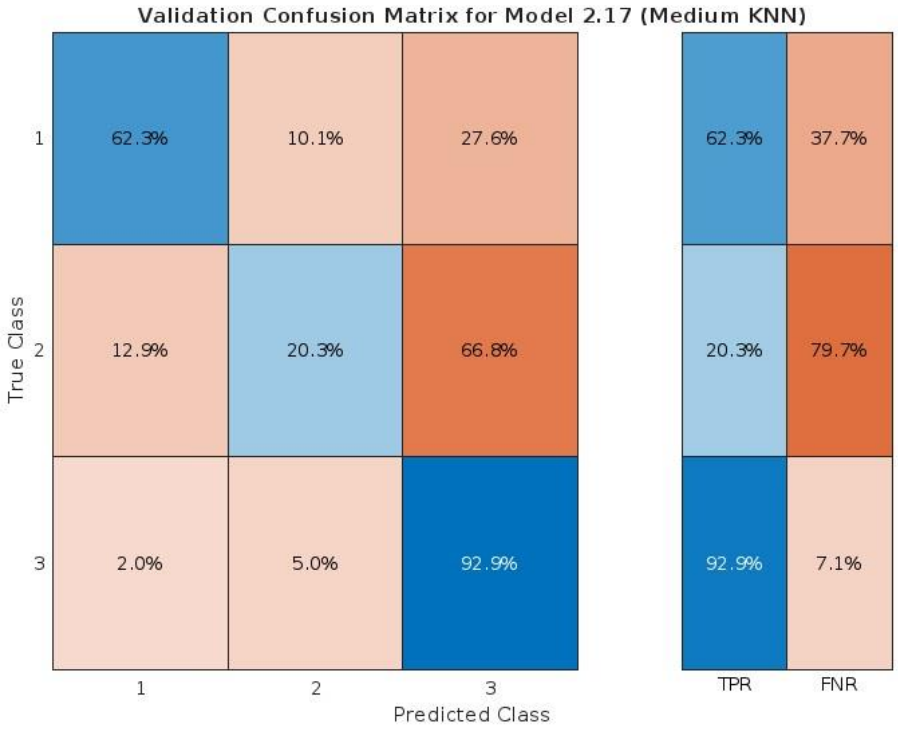


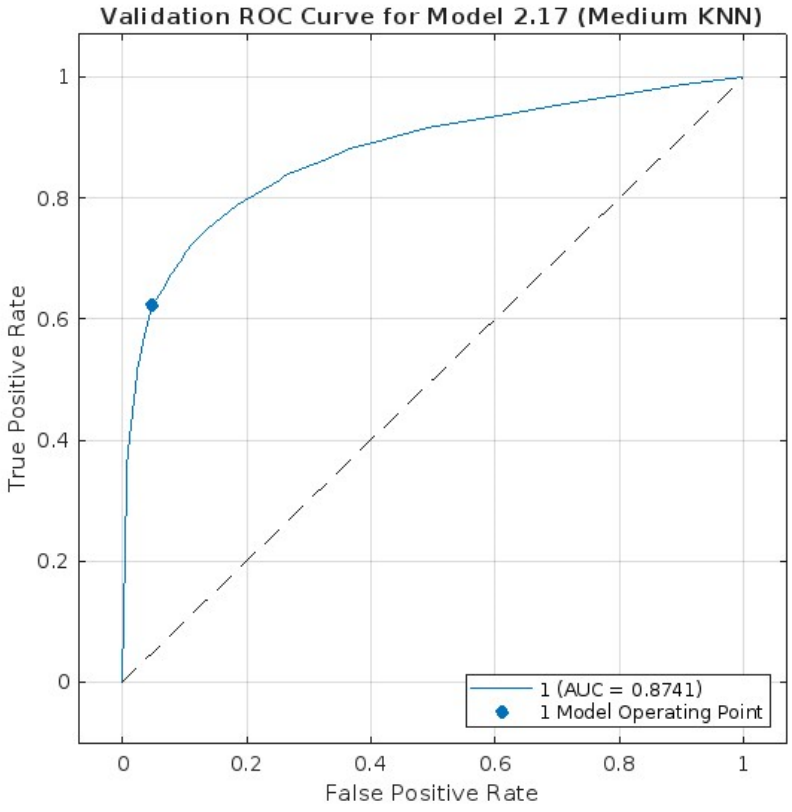
Testing graph


Test Confusion Matrix for Model 2.16 (Fine KNN)
True Class
Predicted Class
1
2
3
58.4%
18.7%
22.9%
19.4%
34.1%
46.4%
8.5%
14.1%
77.3%
58.4%
41.6%
34.1%
65.9%
77.3%
22.7%
TPR
FNR

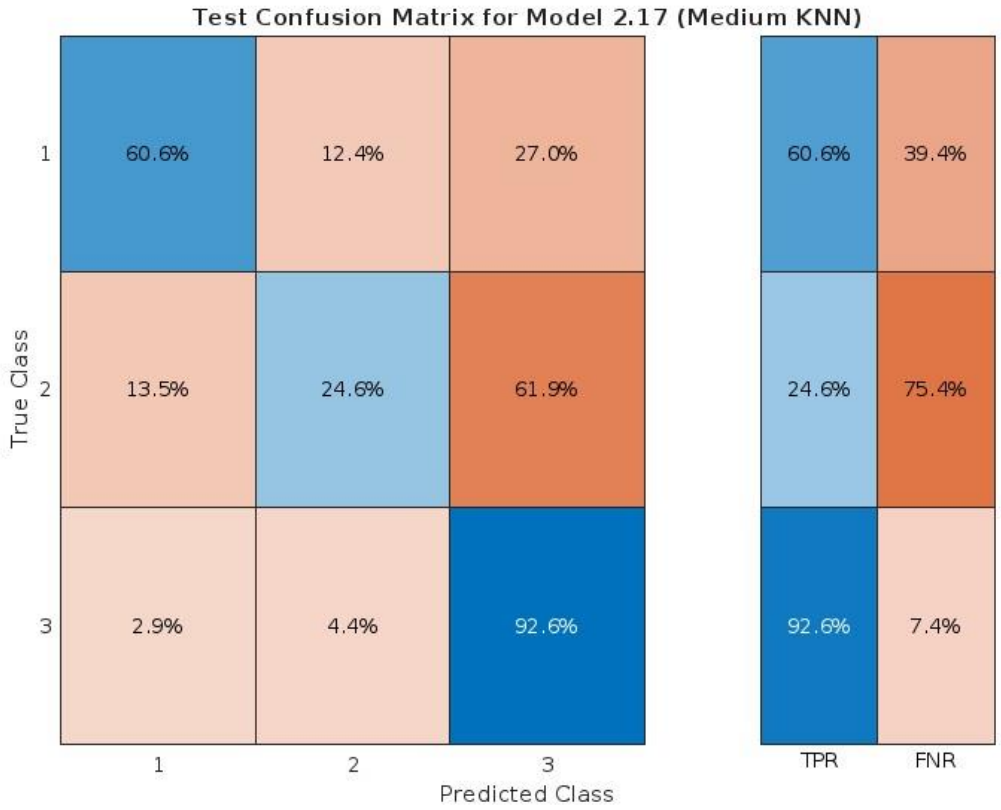

Test ROC Curve for Model 2.16 (Fine KNN)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1 (AUC = 0.7346)
1 Model Operating Point

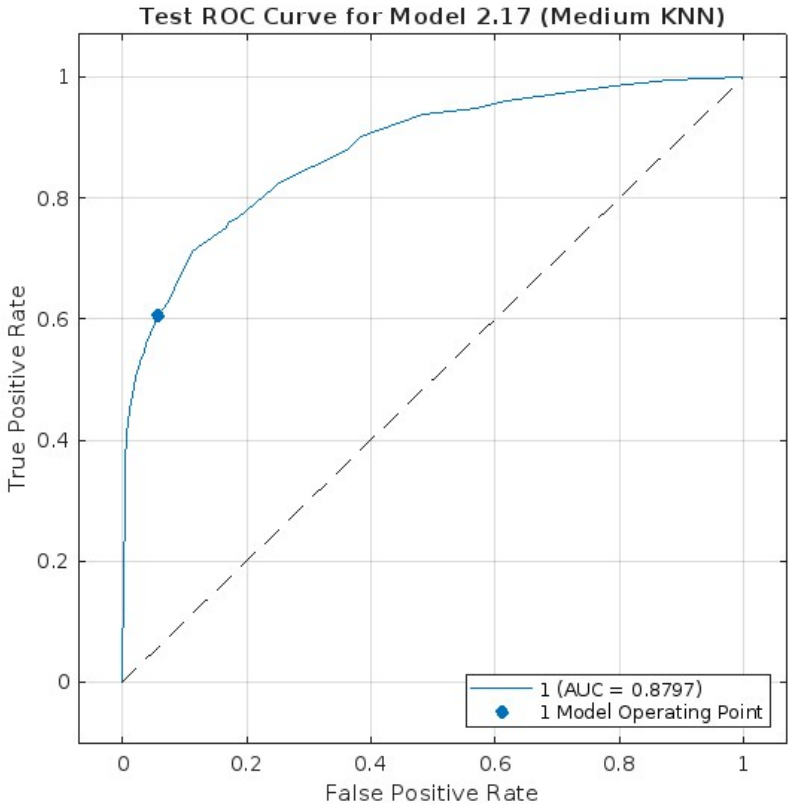
Medium KNN-Training graph



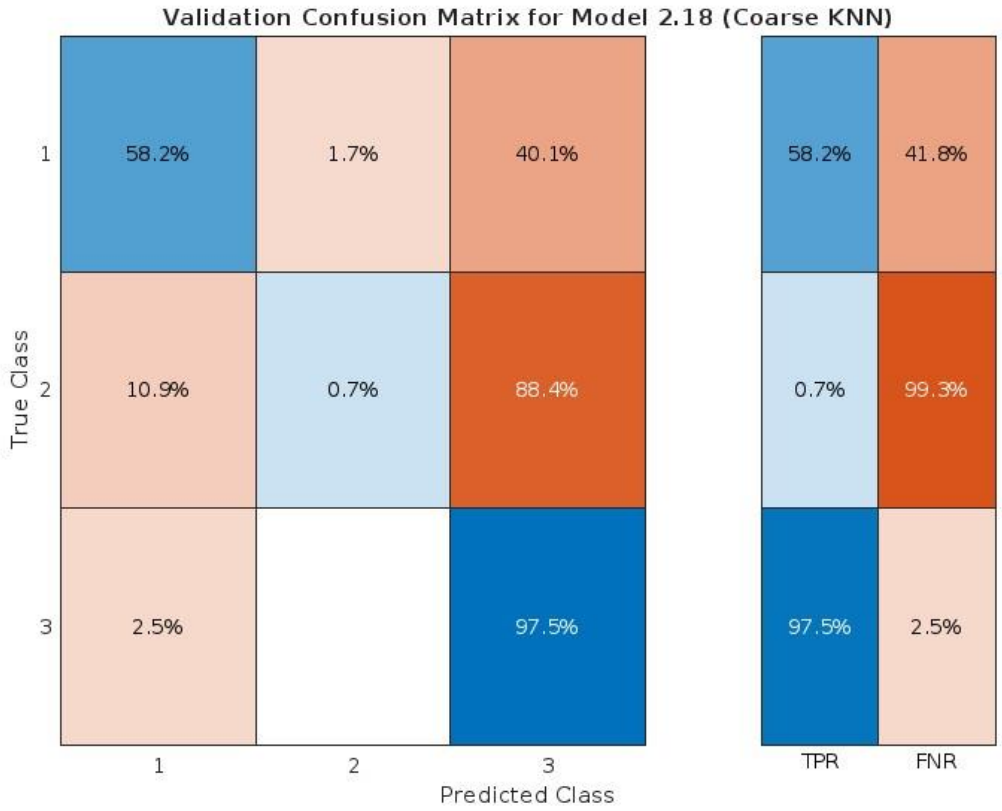


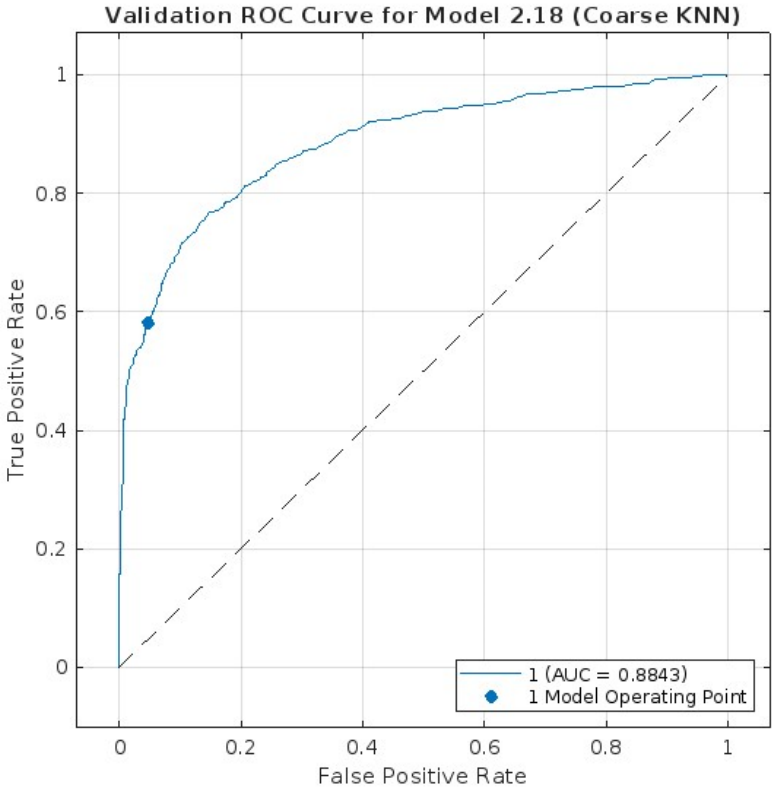
Testing graph



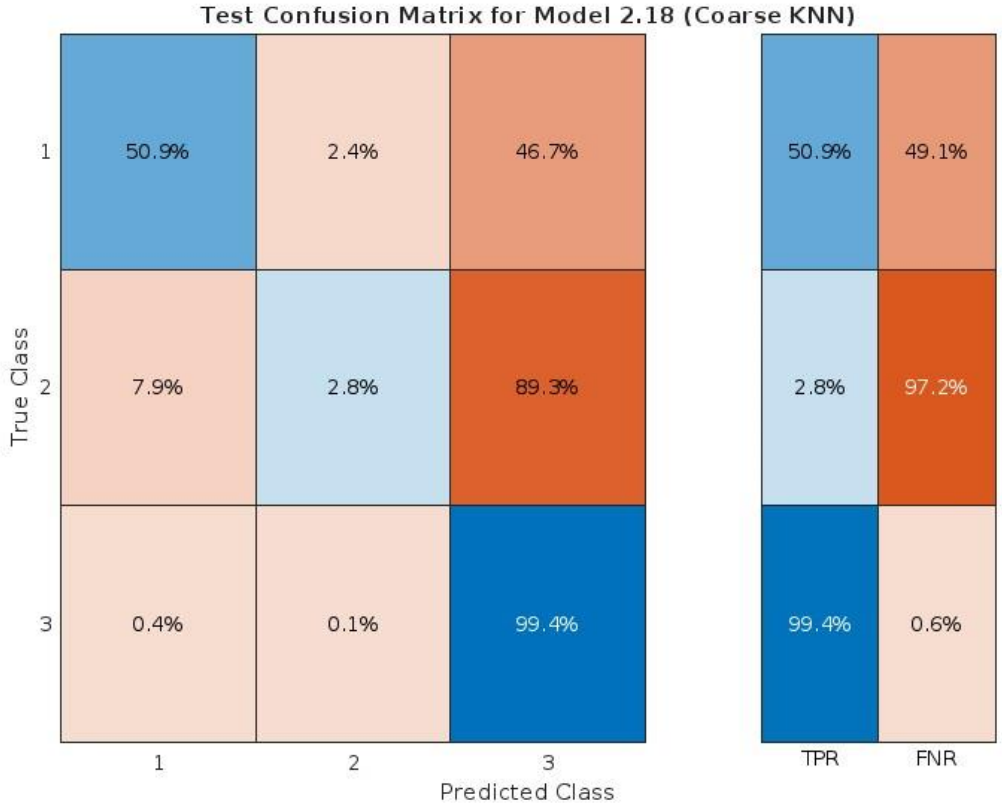


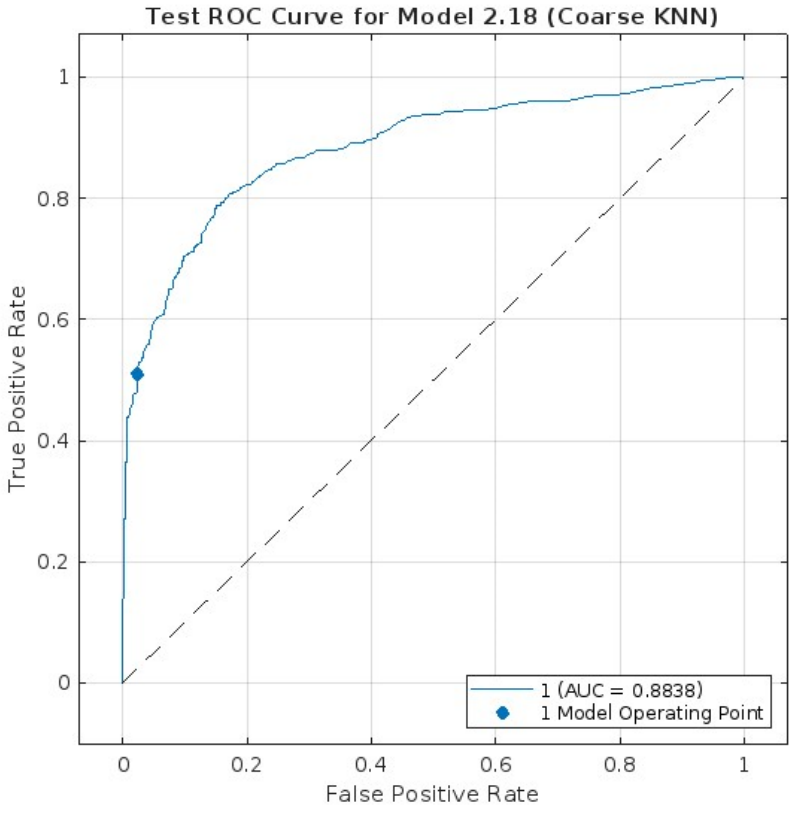
Coarse KNN-Training graph



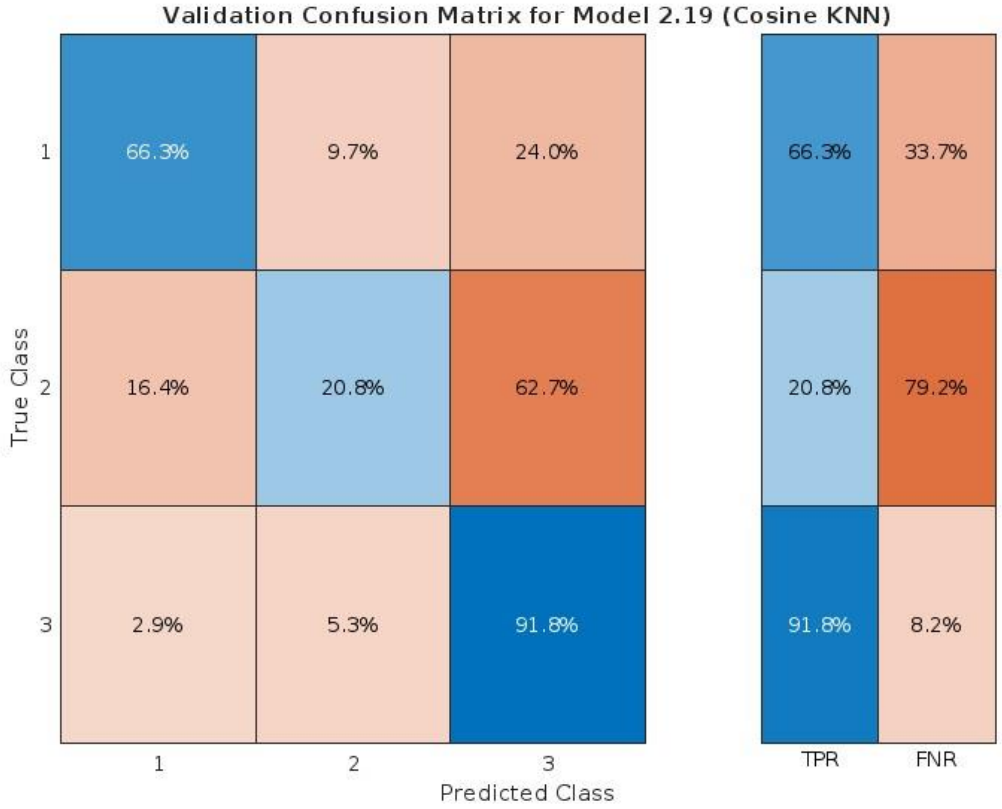


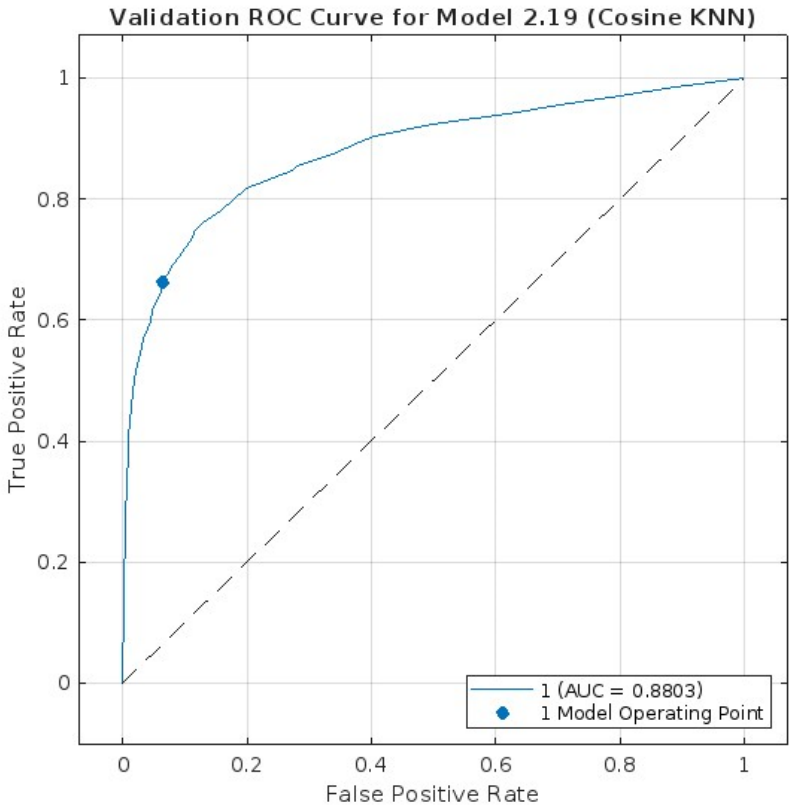
Testing graph


Test Confusion Matrix for Model 2.18 (Coarse KNN)
True Class
Predicted Class
1
2
3
50.9%
2.4%
46.7%
7.9%
2.8%
89.3%
0.4%
0.1%
99.4%
50.9%
49.1%
2.8%
97.2%
99.4%
0.6%
TPR
FNR

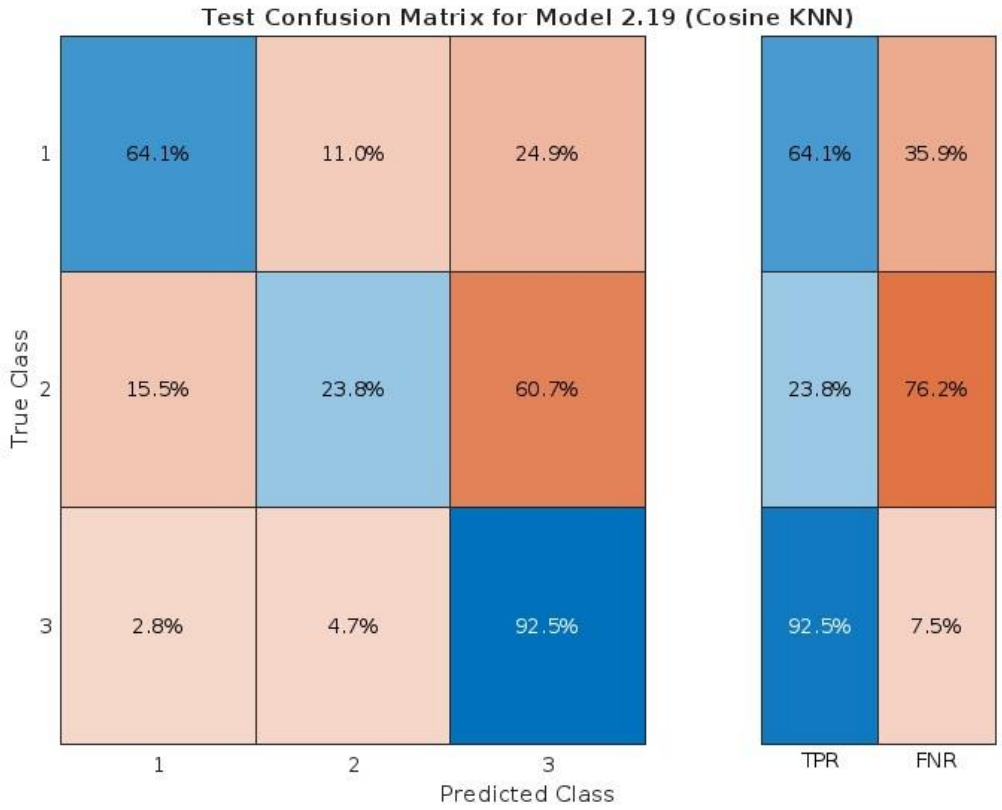

Test ROC Curve for Model 2.18 (Coarse KNN)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1 (AUC = 0.8838)
1 Model Operating Point

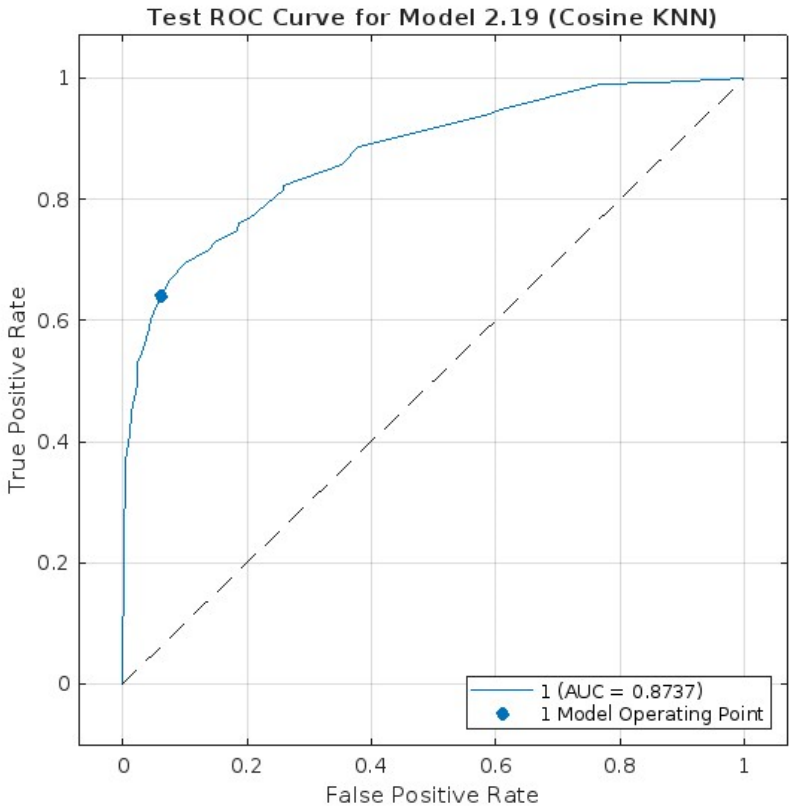
Cosine KNN-training graph



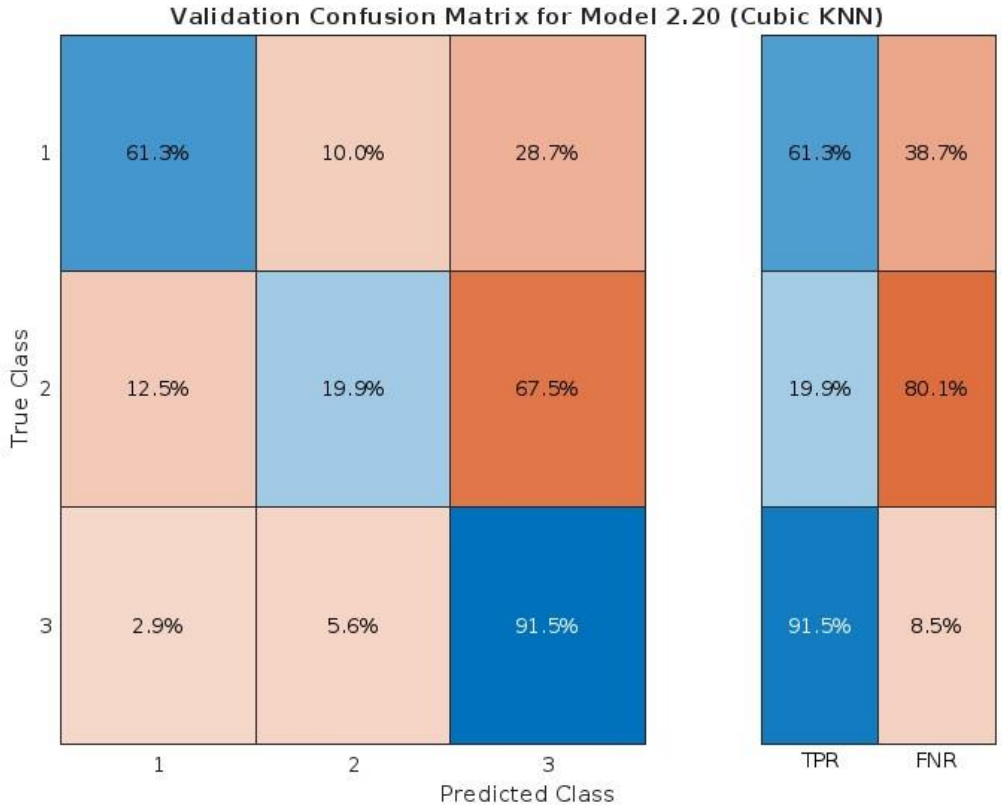


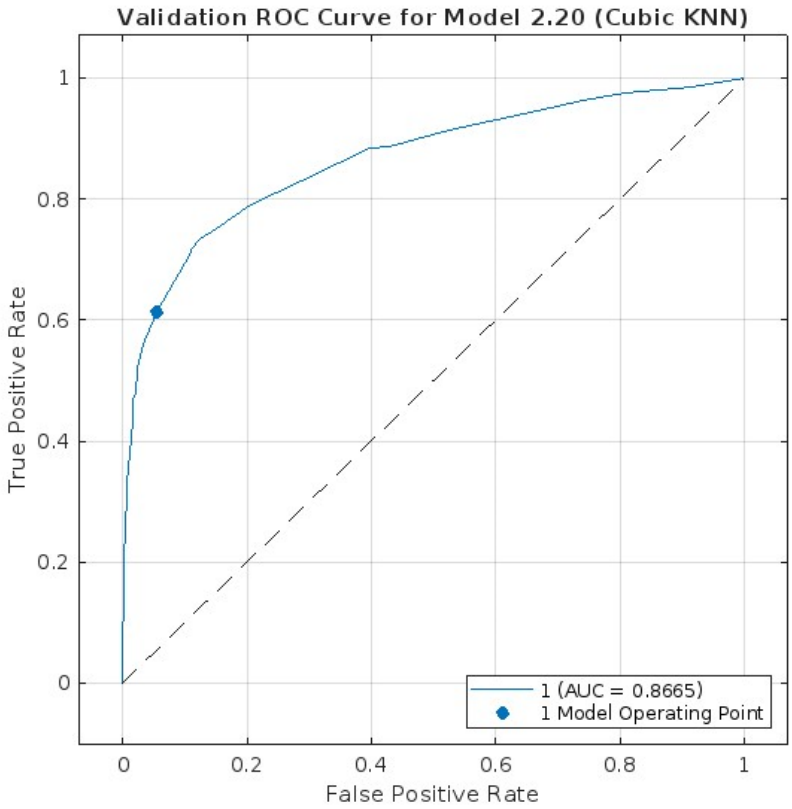
Testing graph



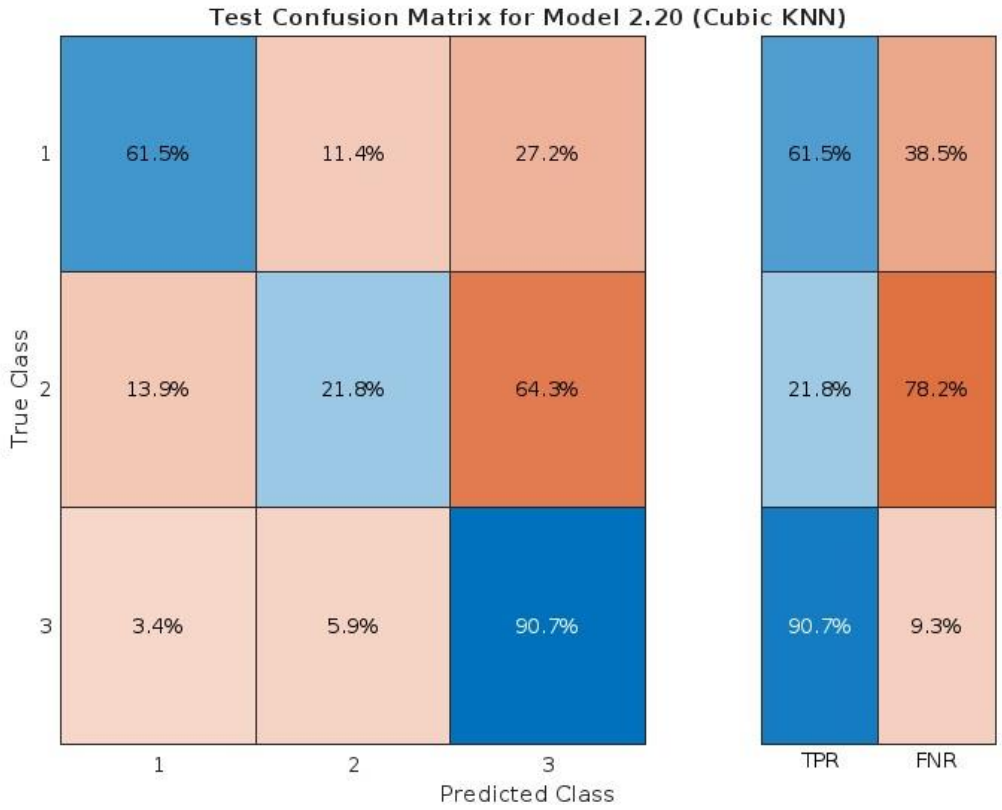


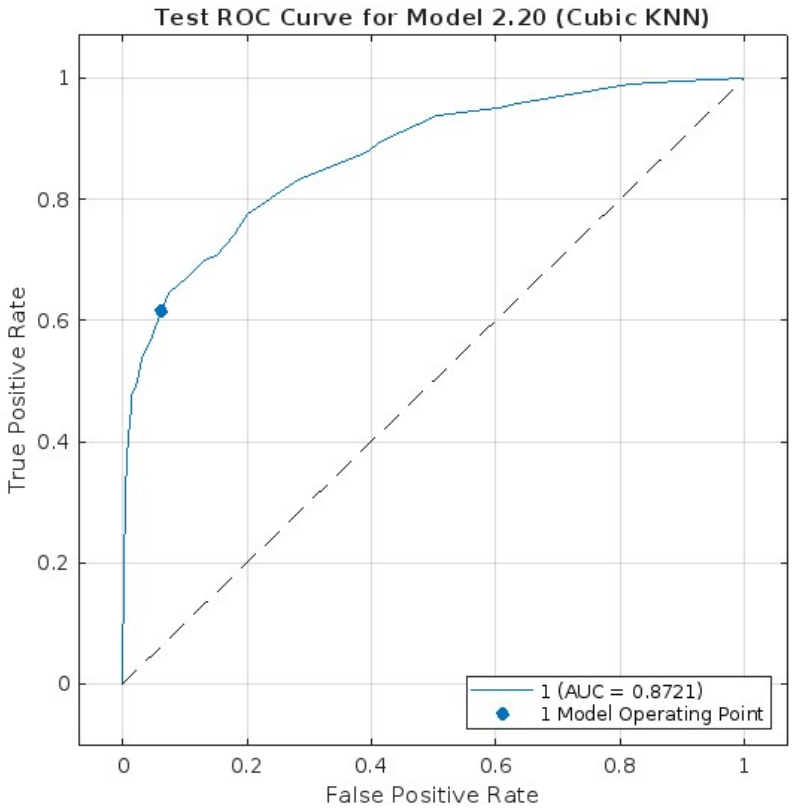
Cubic KNN-training graph



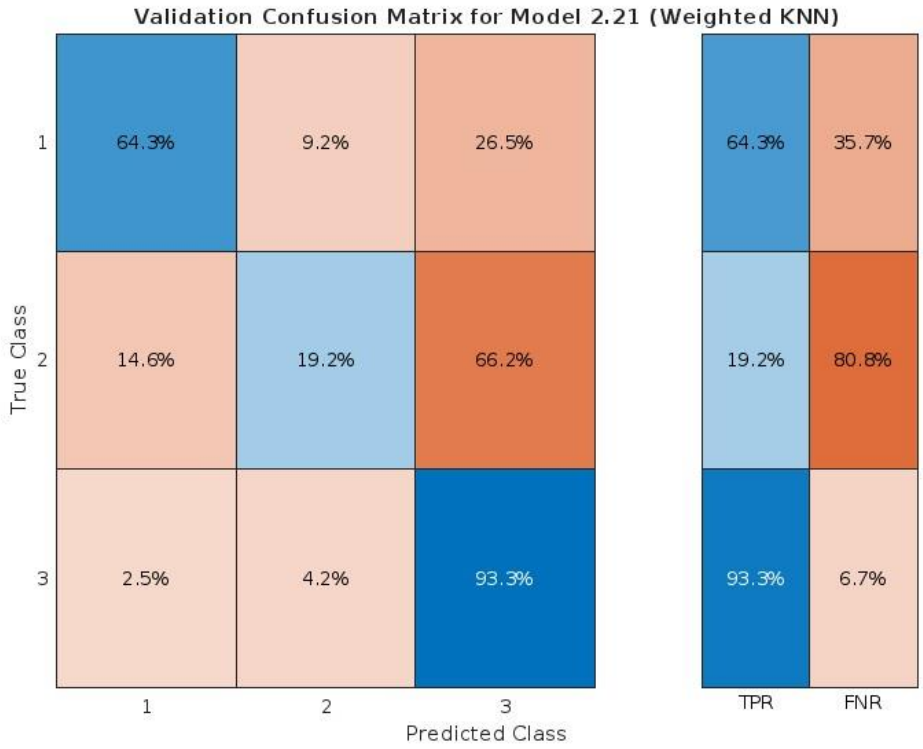


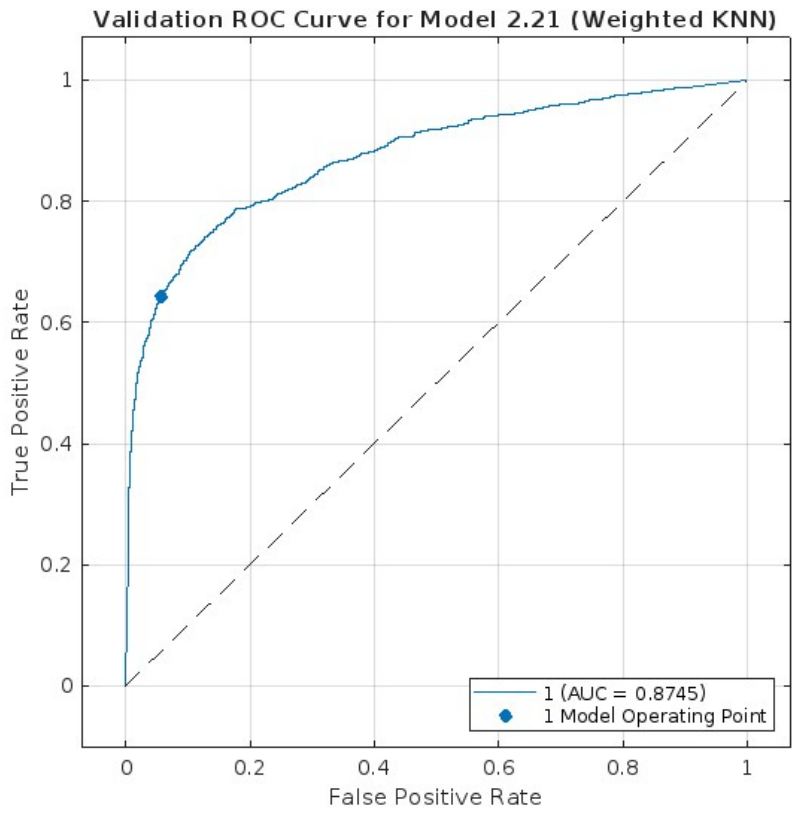
Testing graph



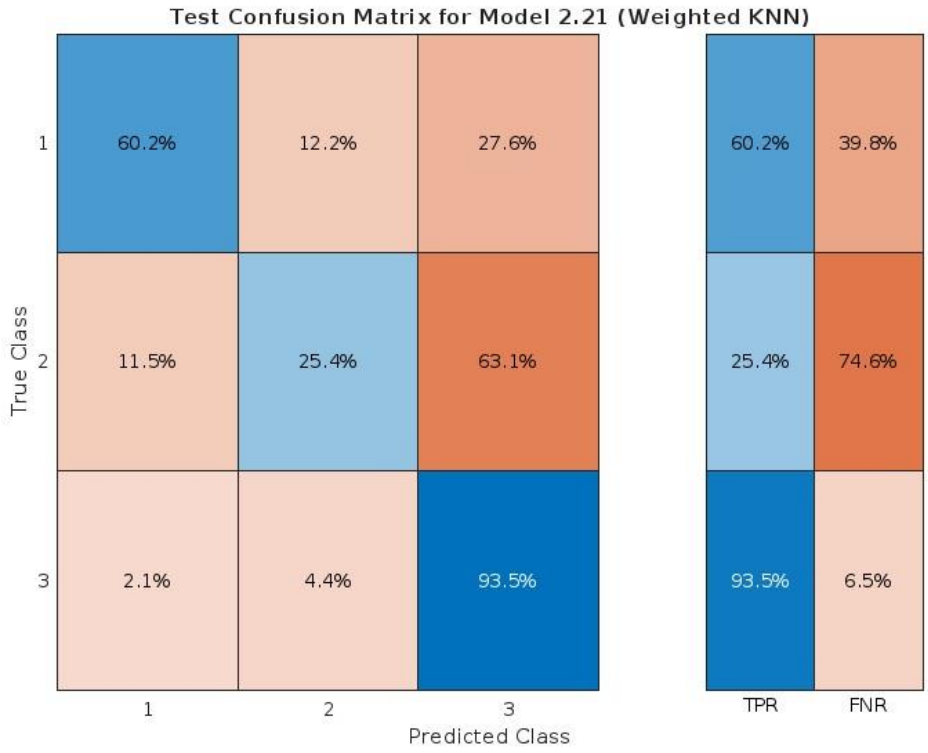


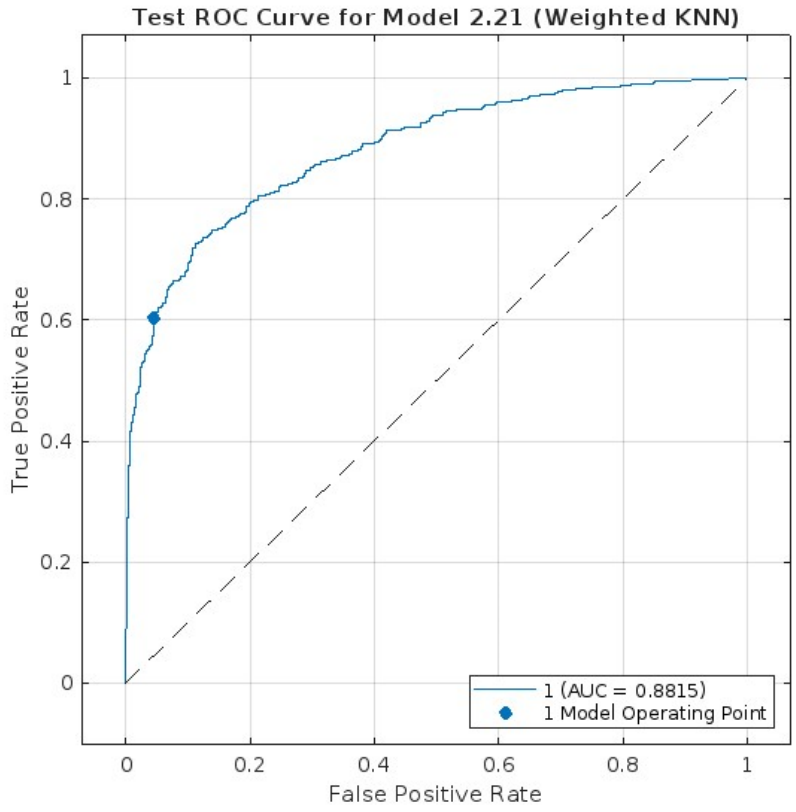
Weighted KNN



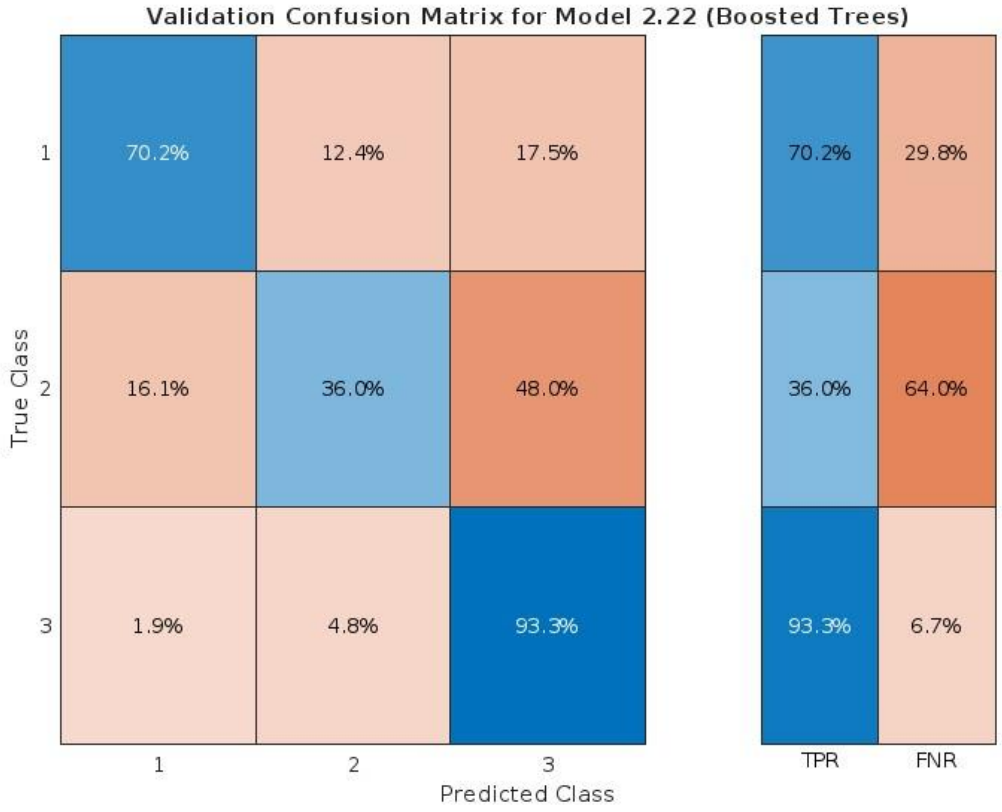


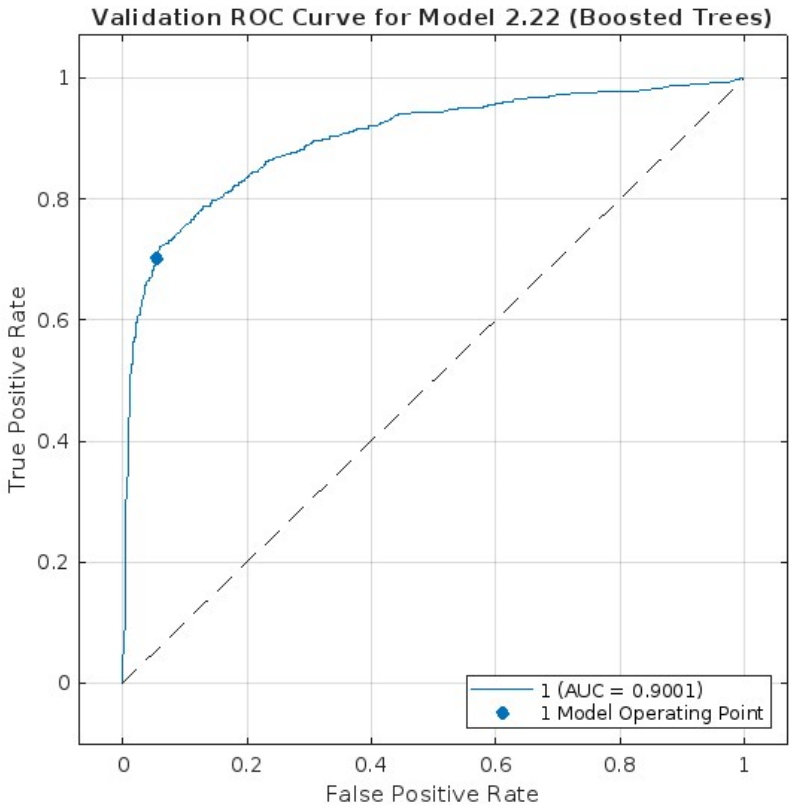
Testing graph



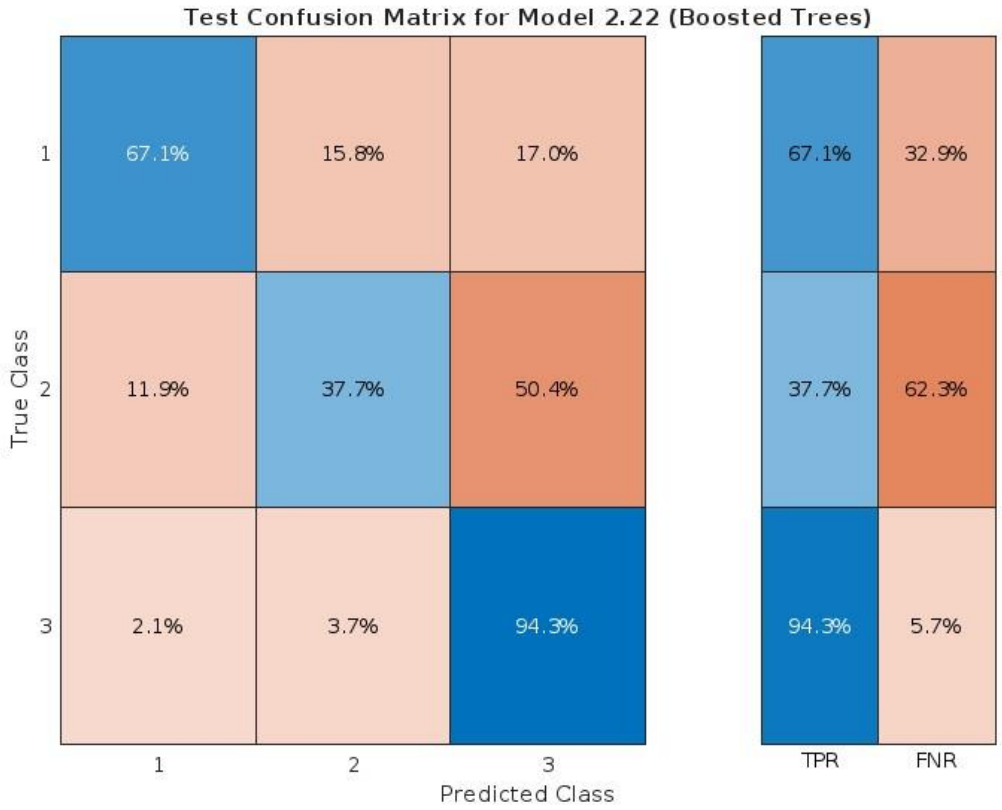


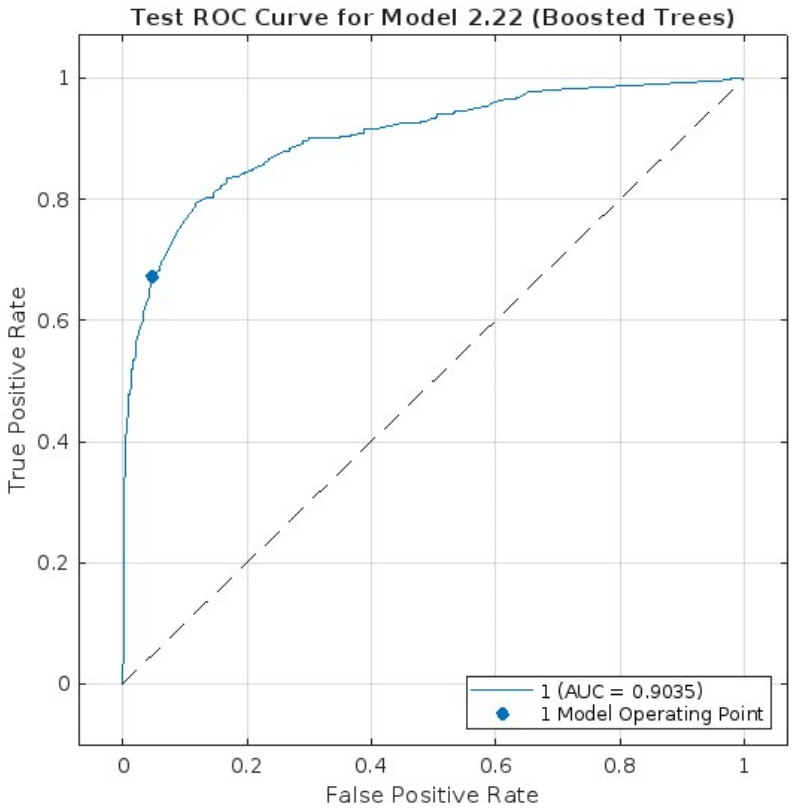
Boosted trees-Training graph



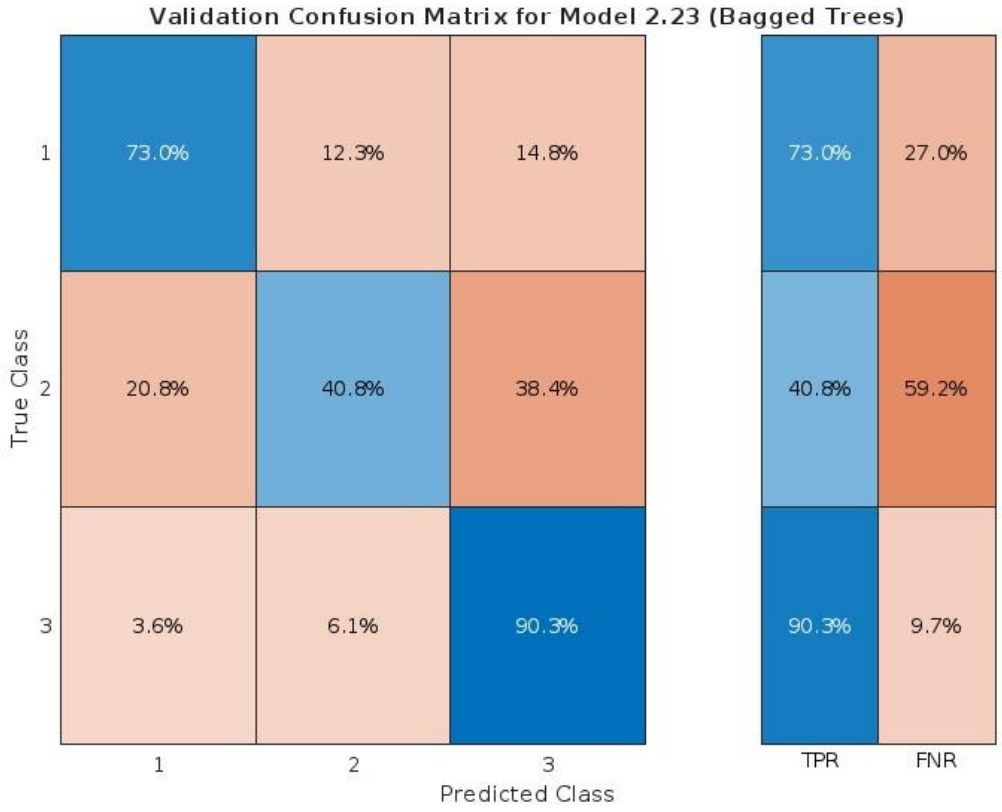


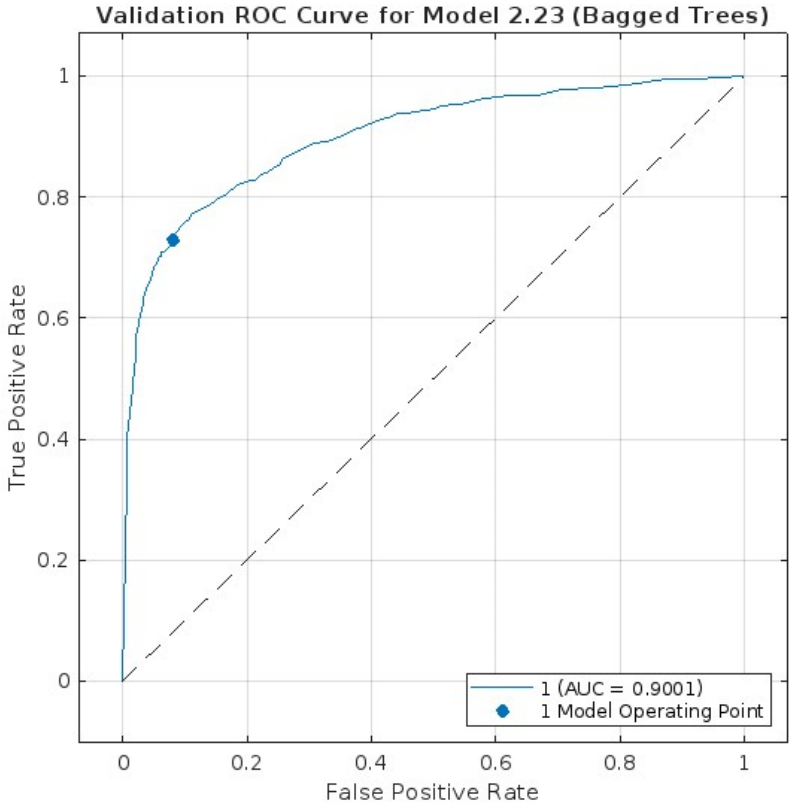
Testing graph



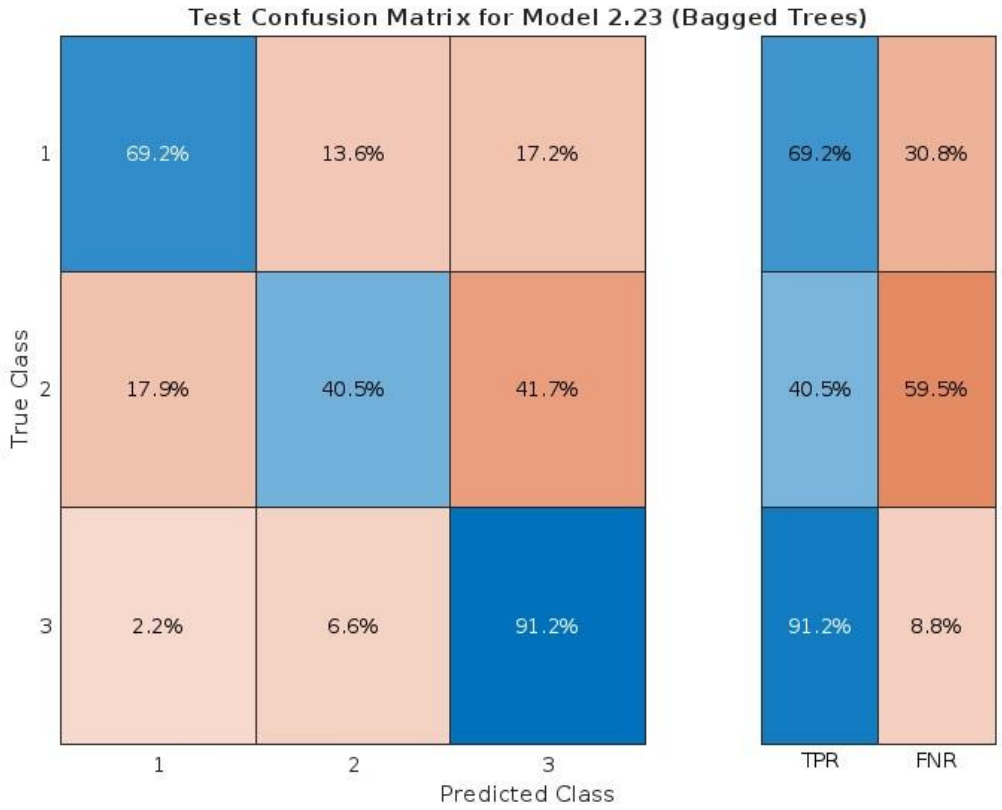


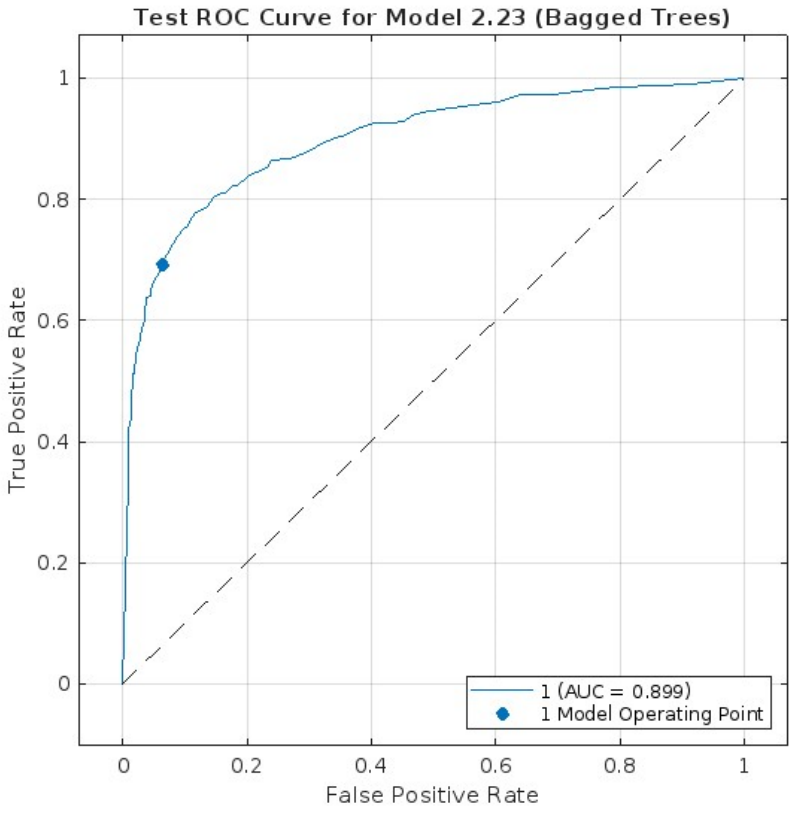
Bagged trees – Training graphs



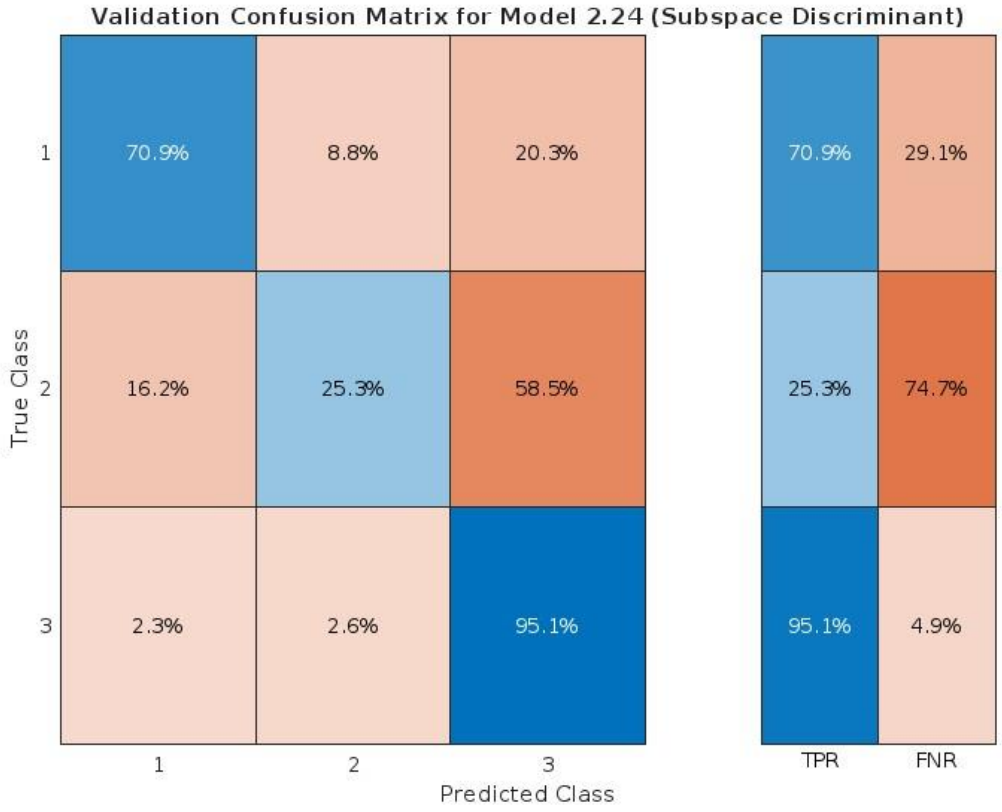


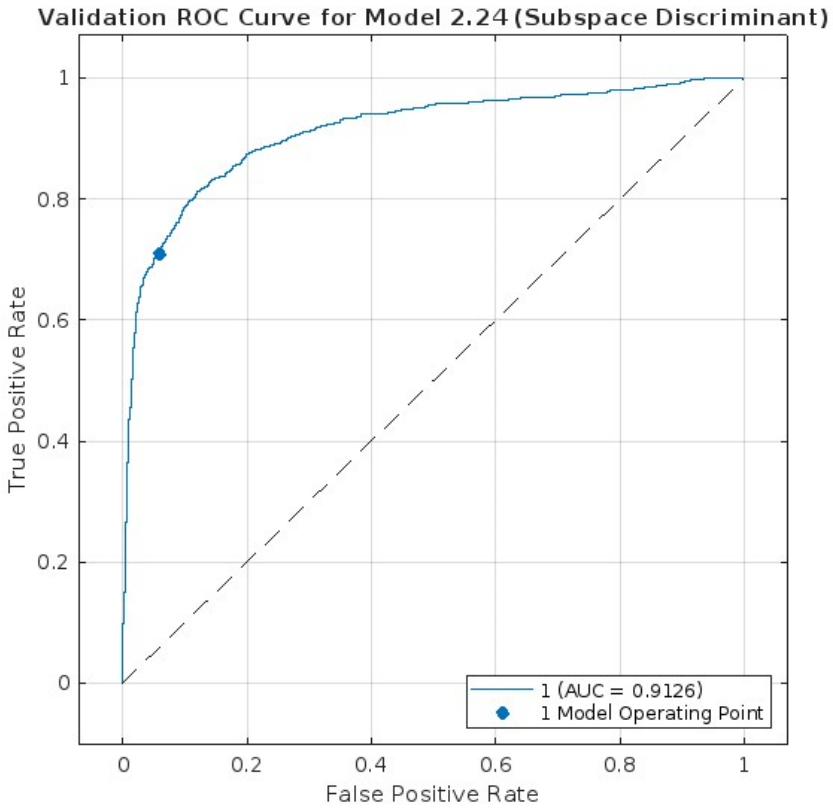
Testing graphs



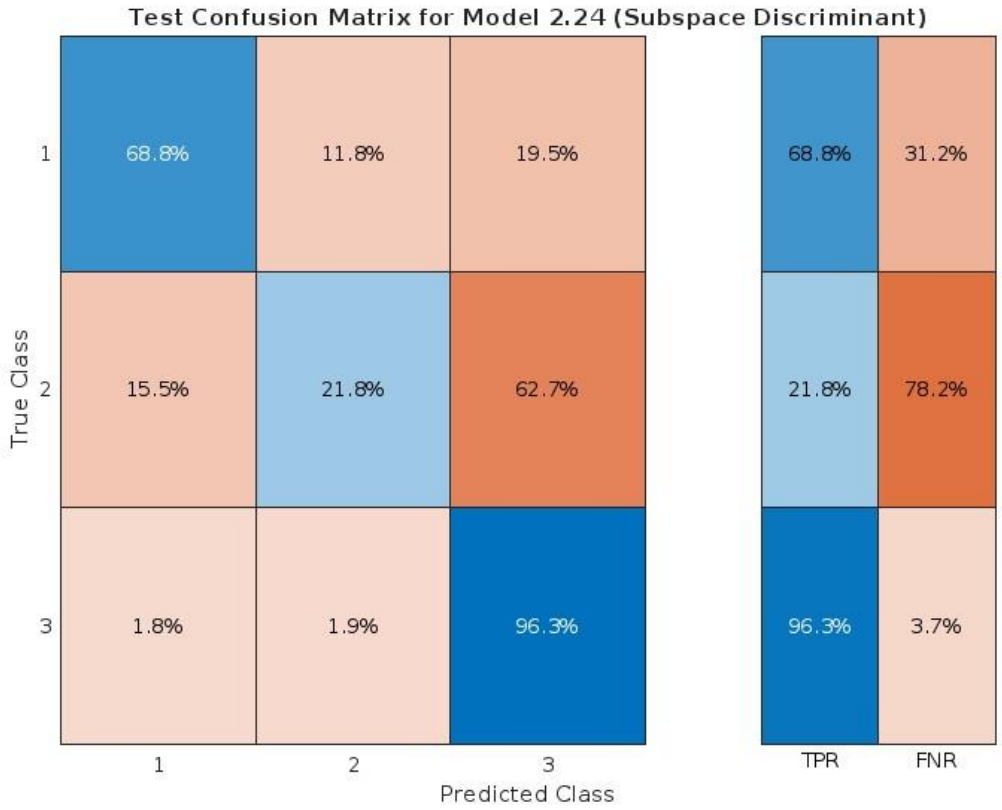


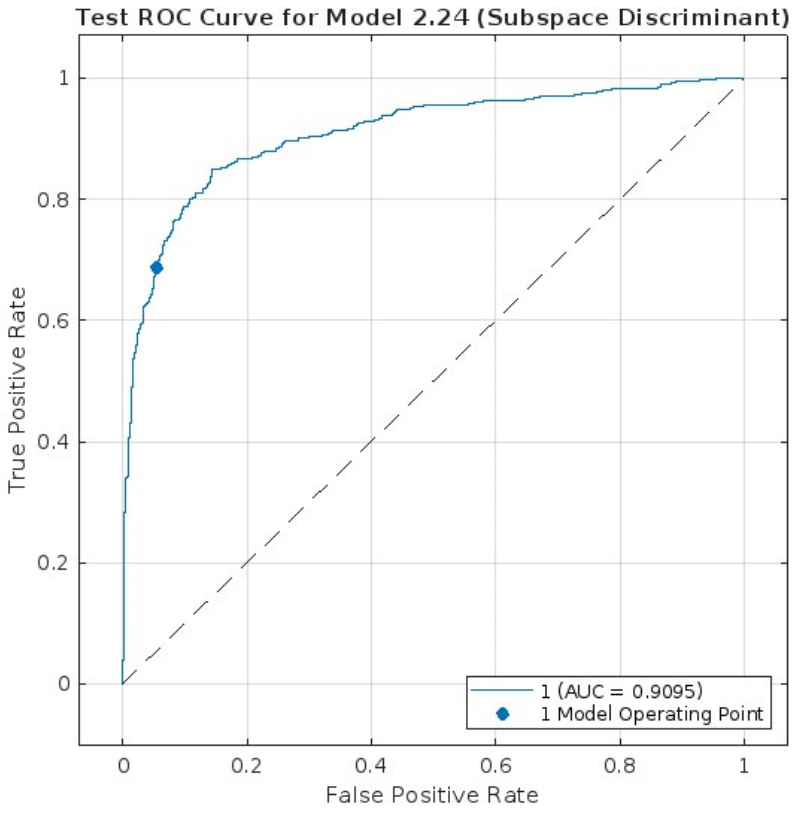
Subspace disciminant-Training graph



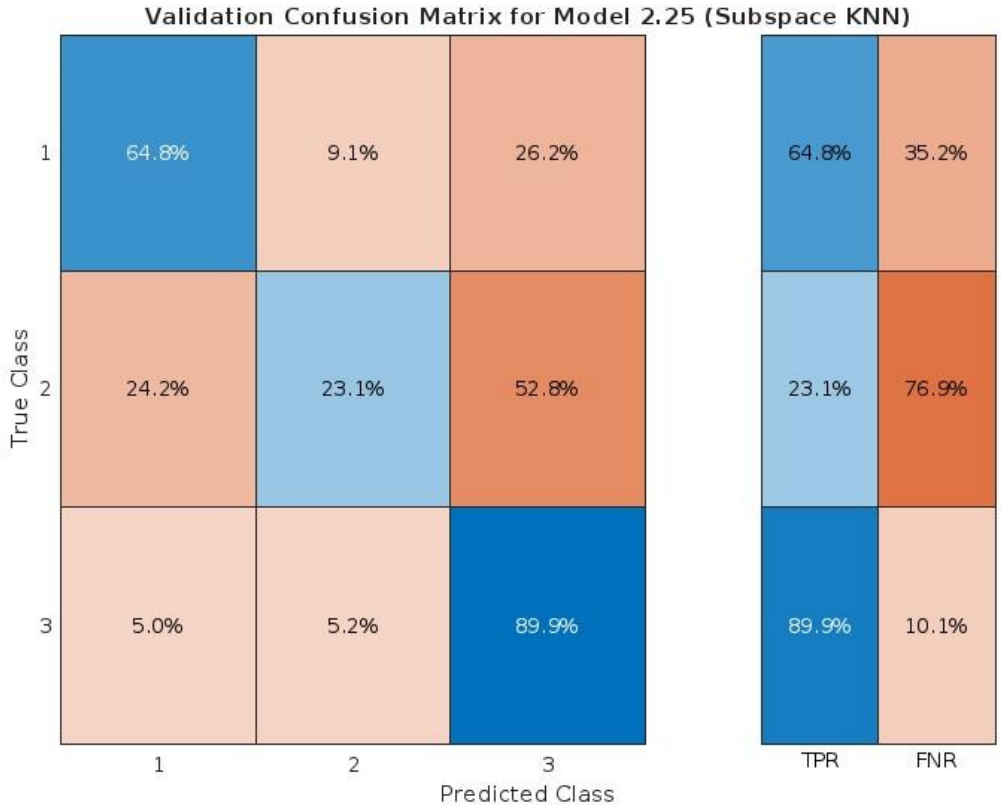


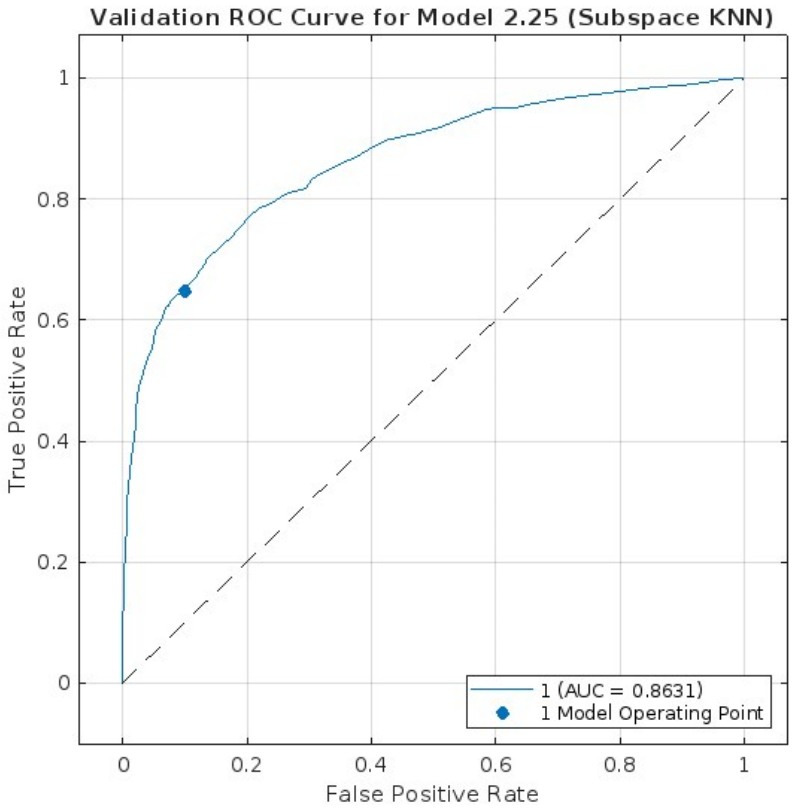
Testing graph



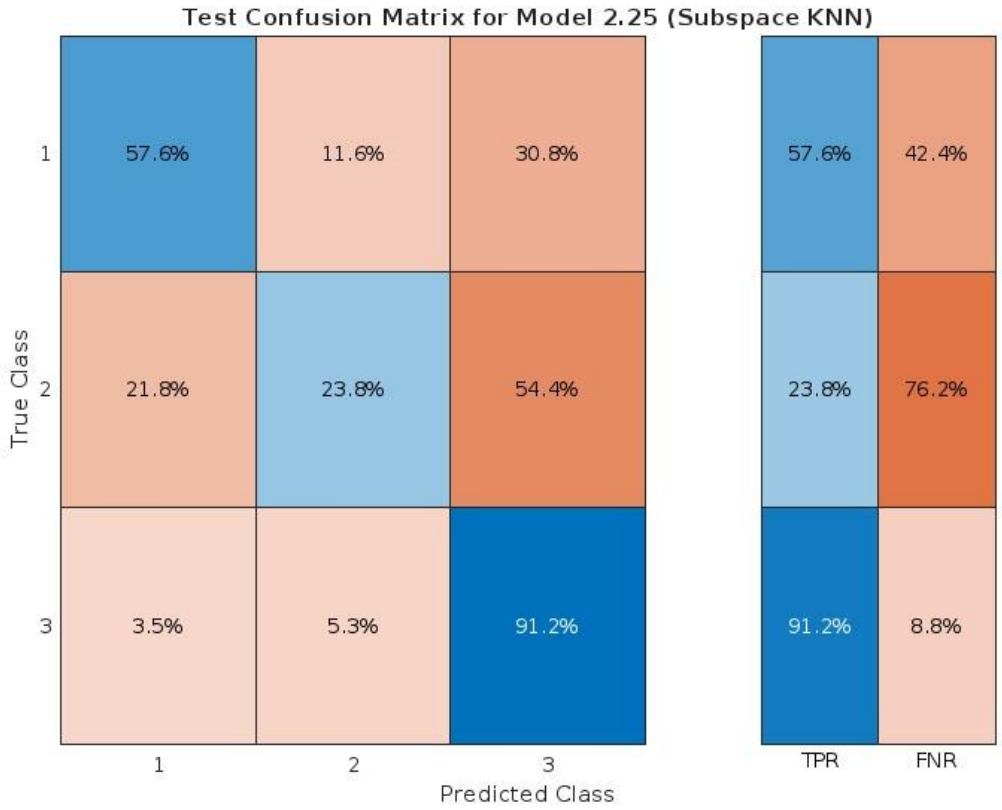


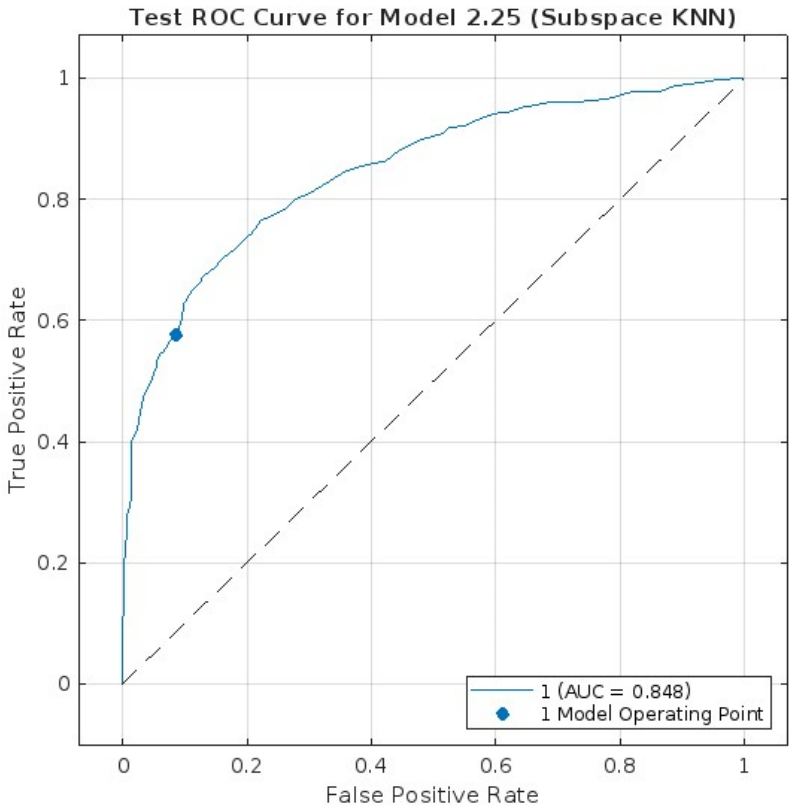
Subspace KNN- Training graph



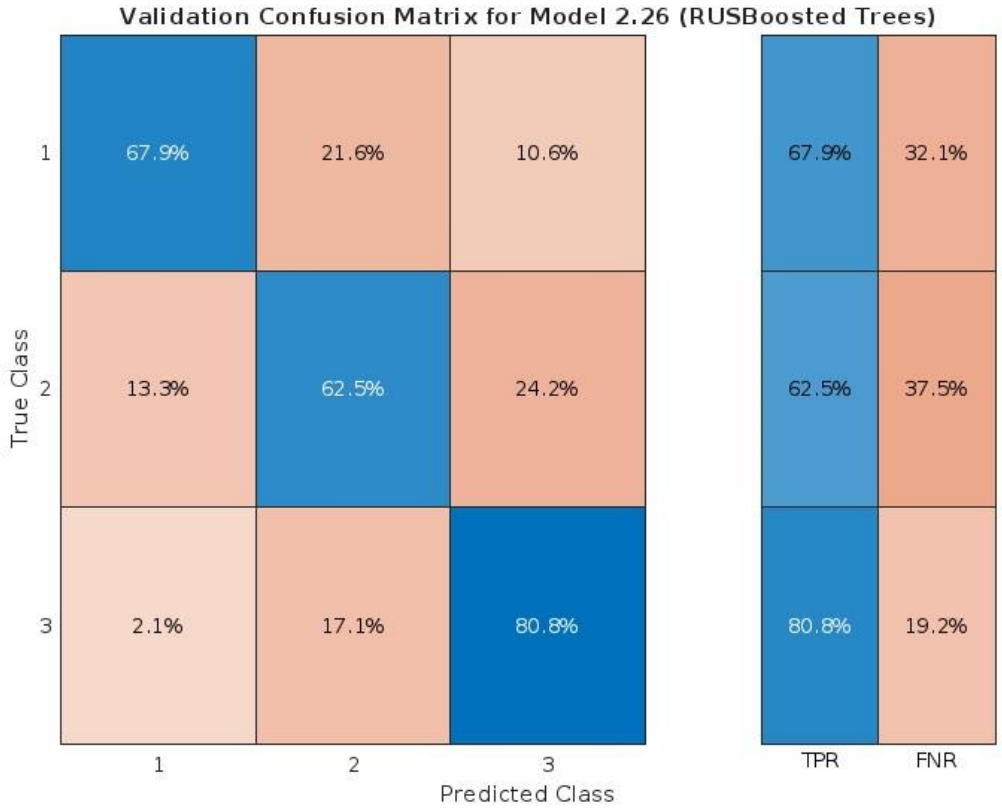


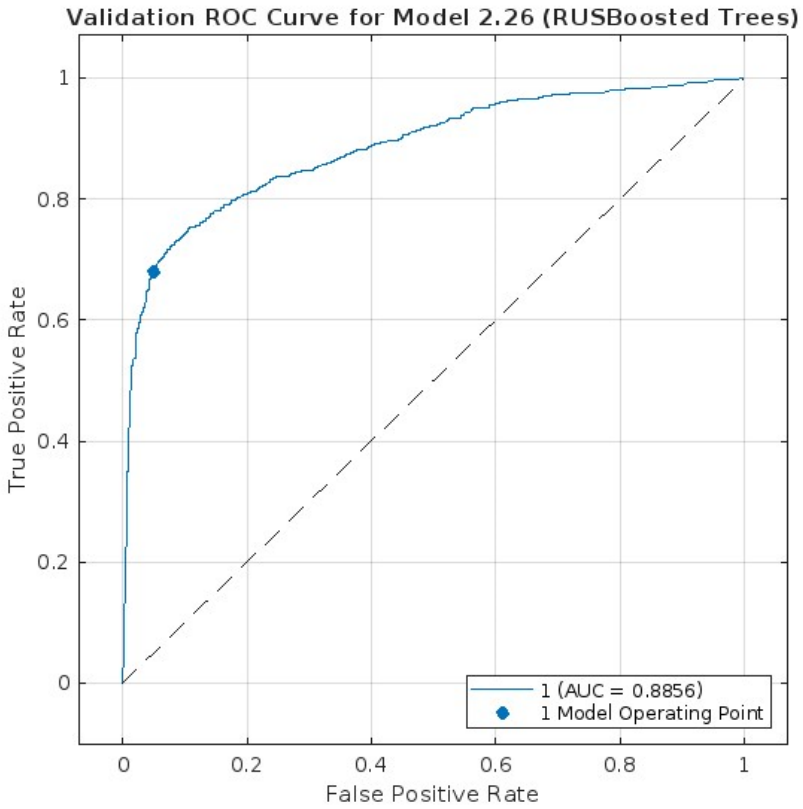
Testing graph



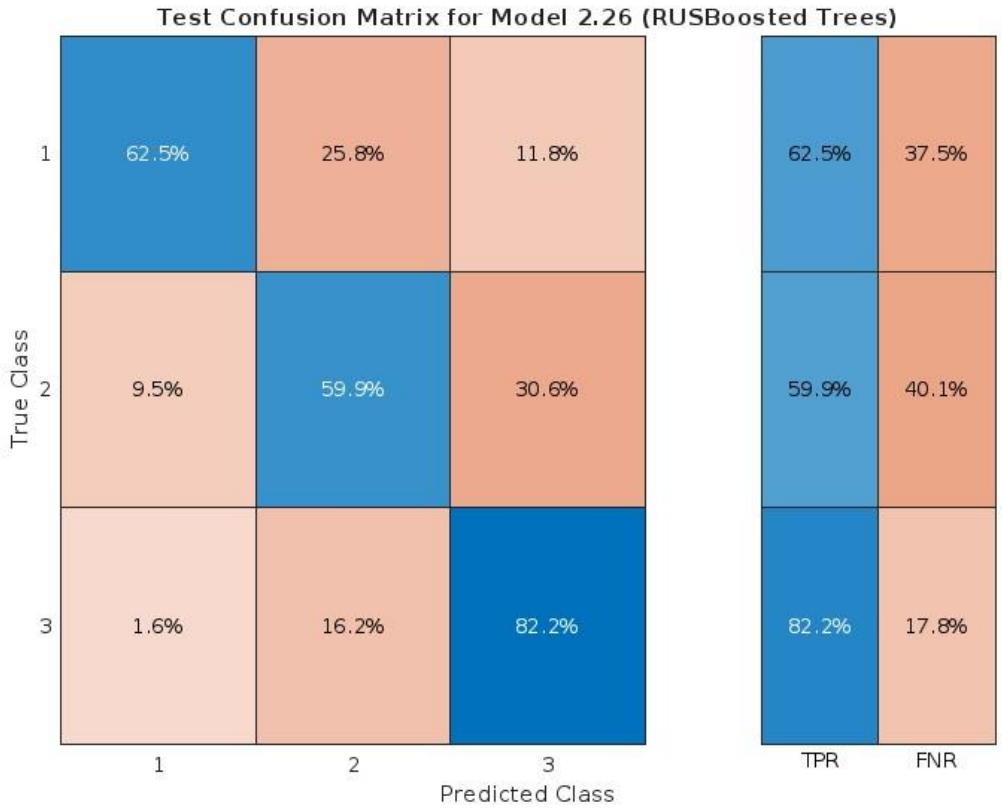


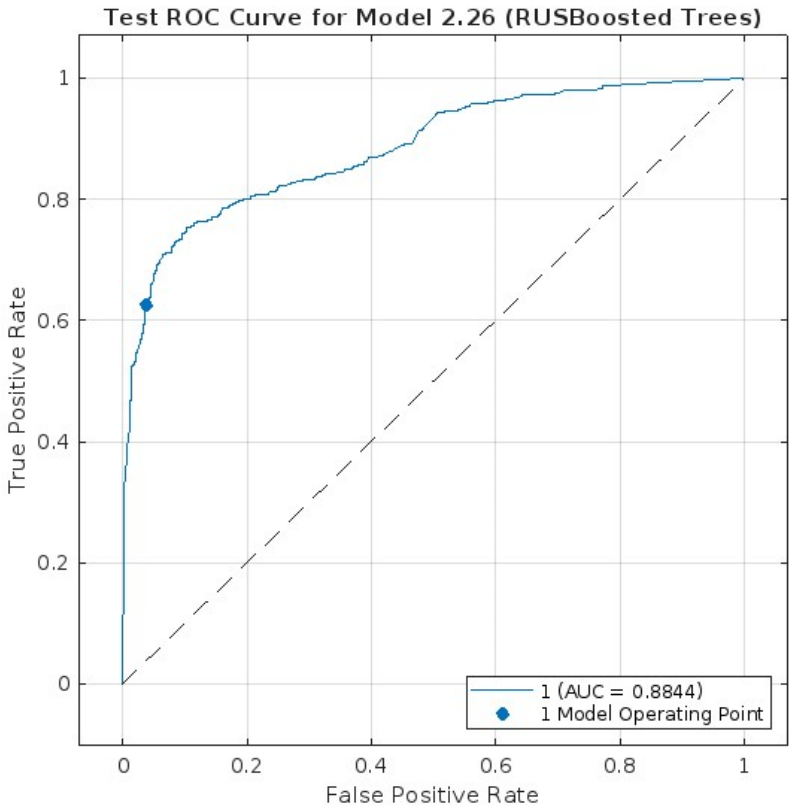
RUS Boosted Trees – Training graph



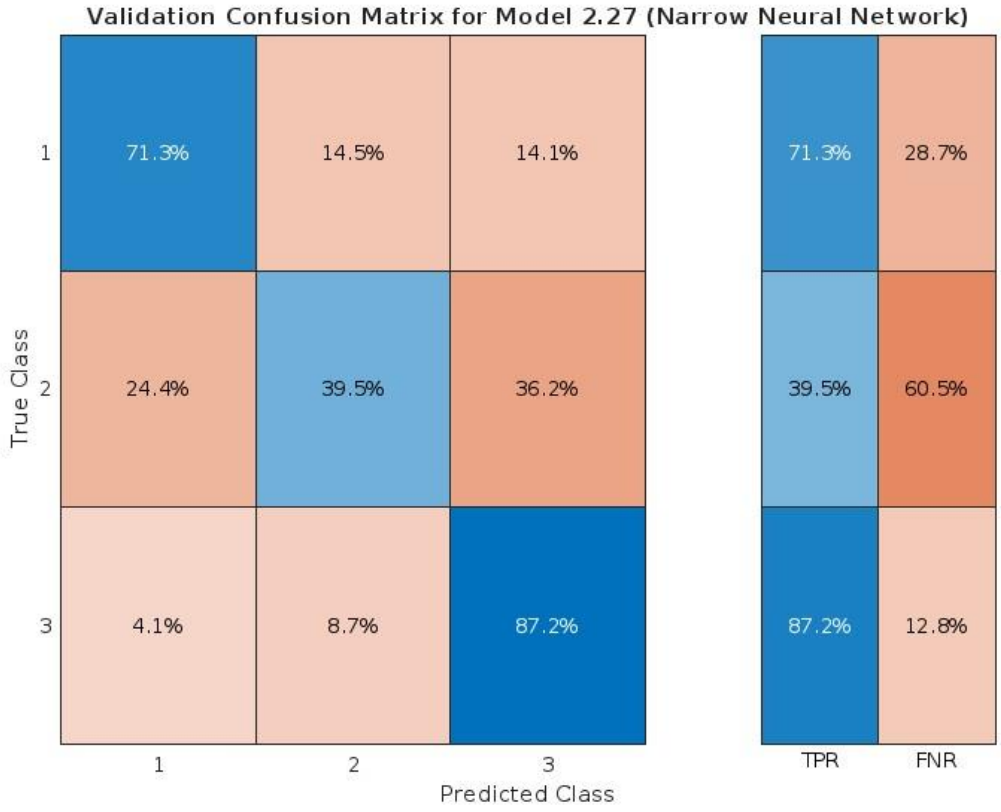


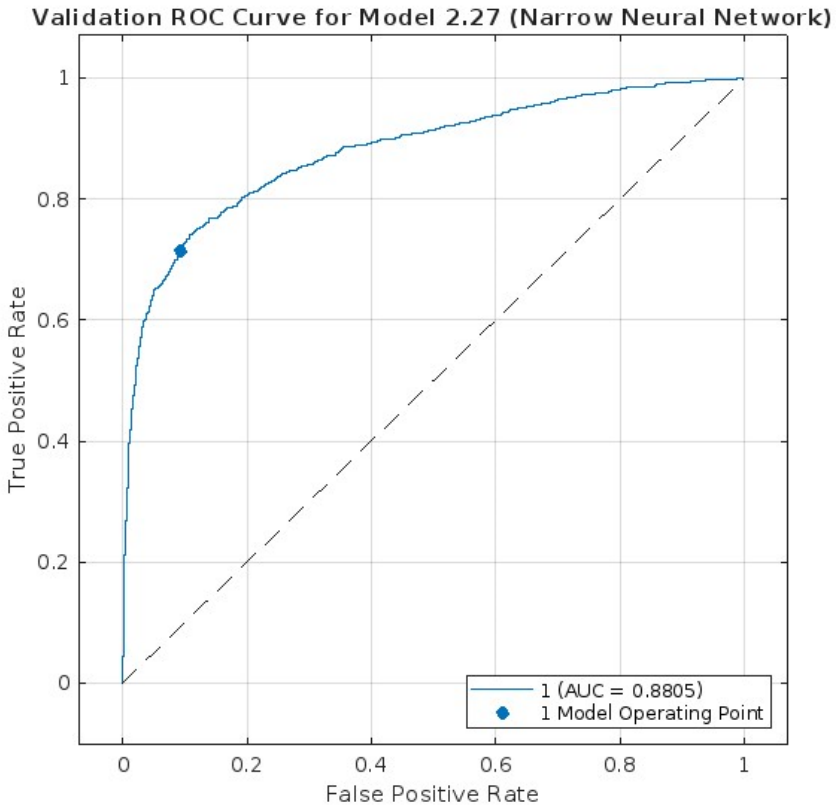
Testing graph



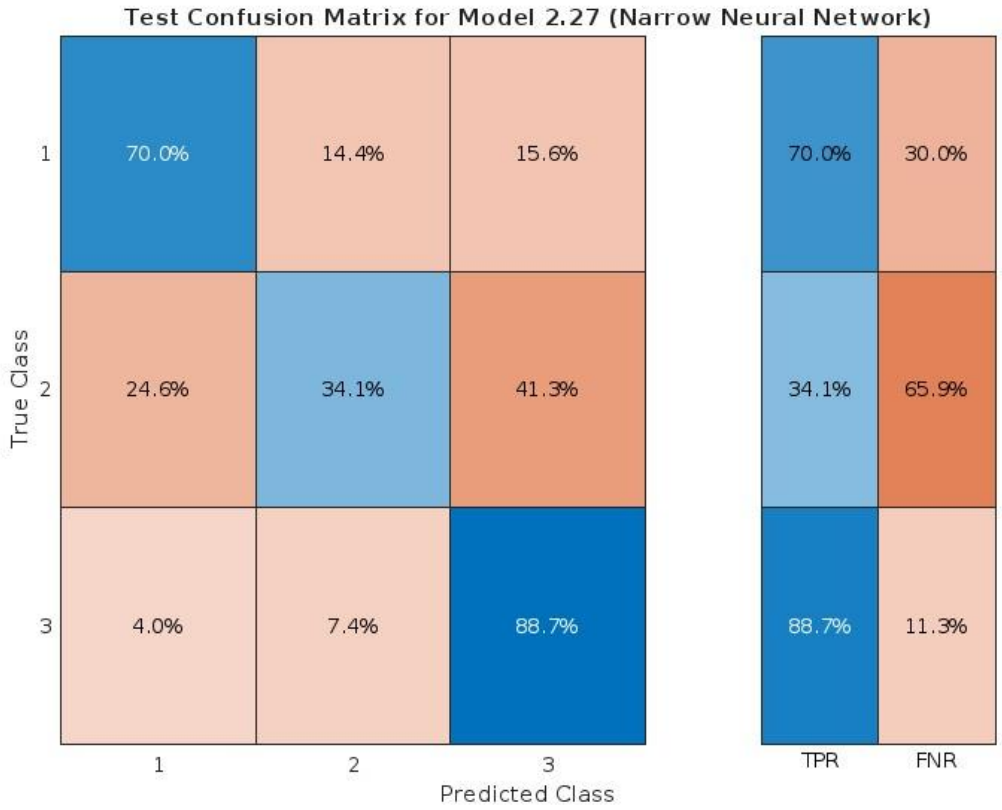


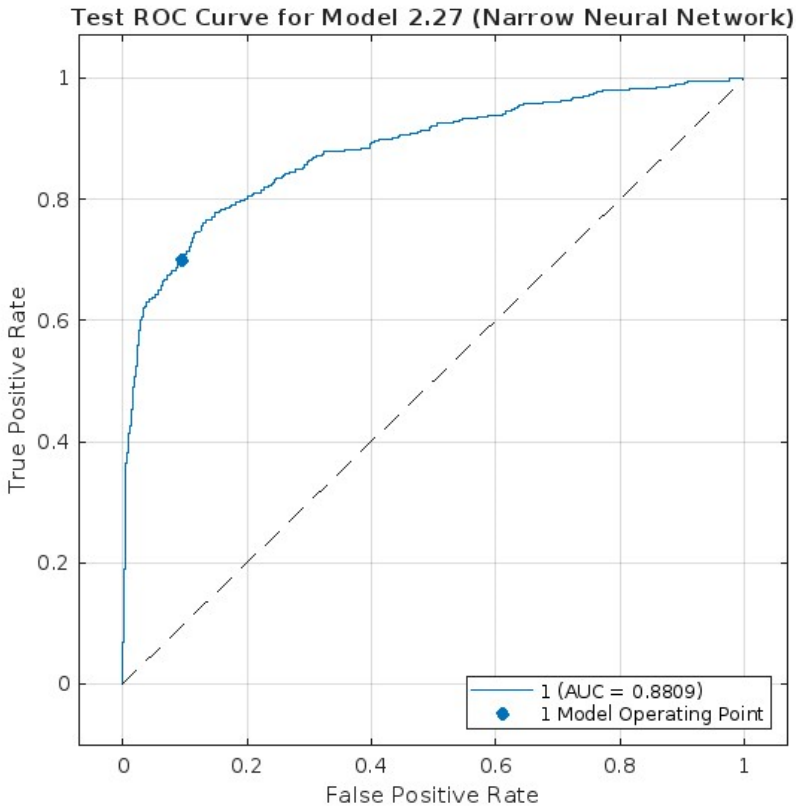
Narrow Neural Network - Training graph



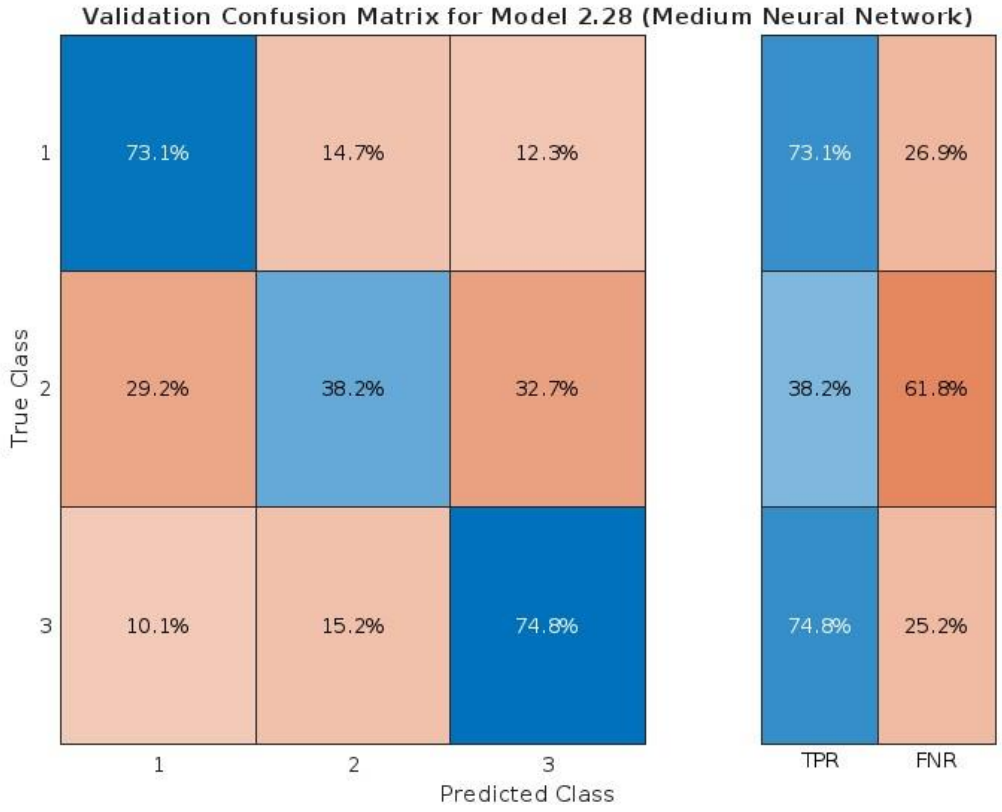


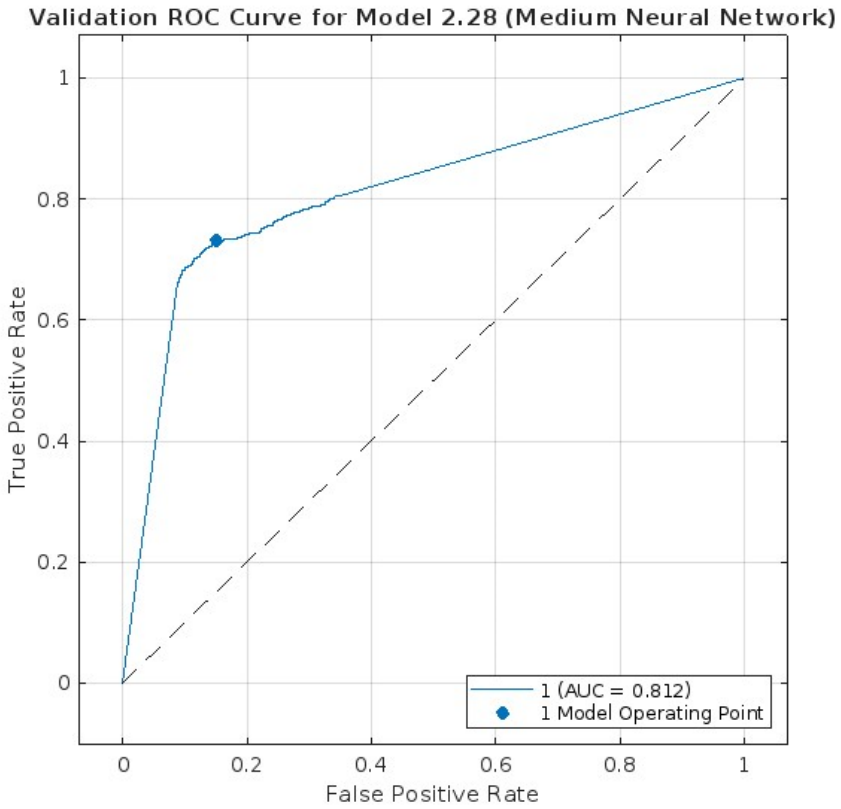
Testing graph



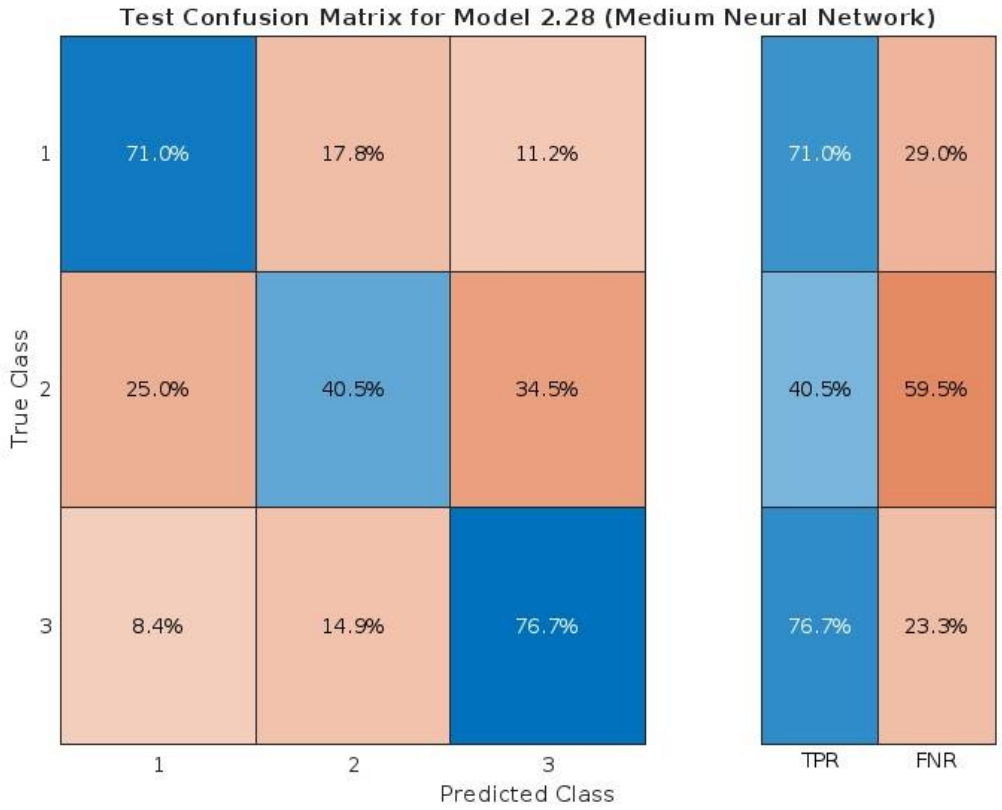


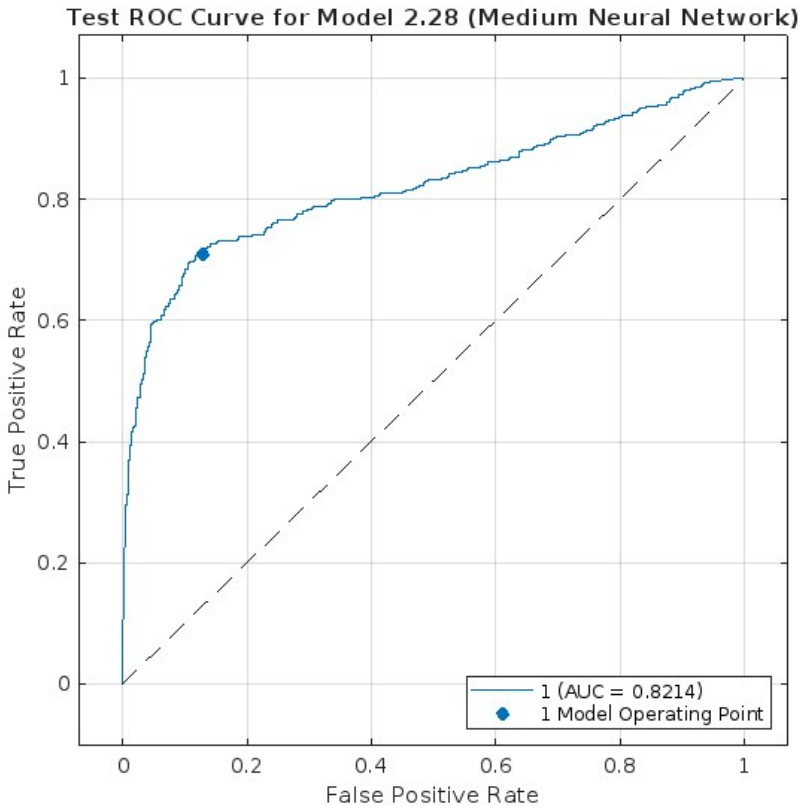
Medium Neural Network - Training graph



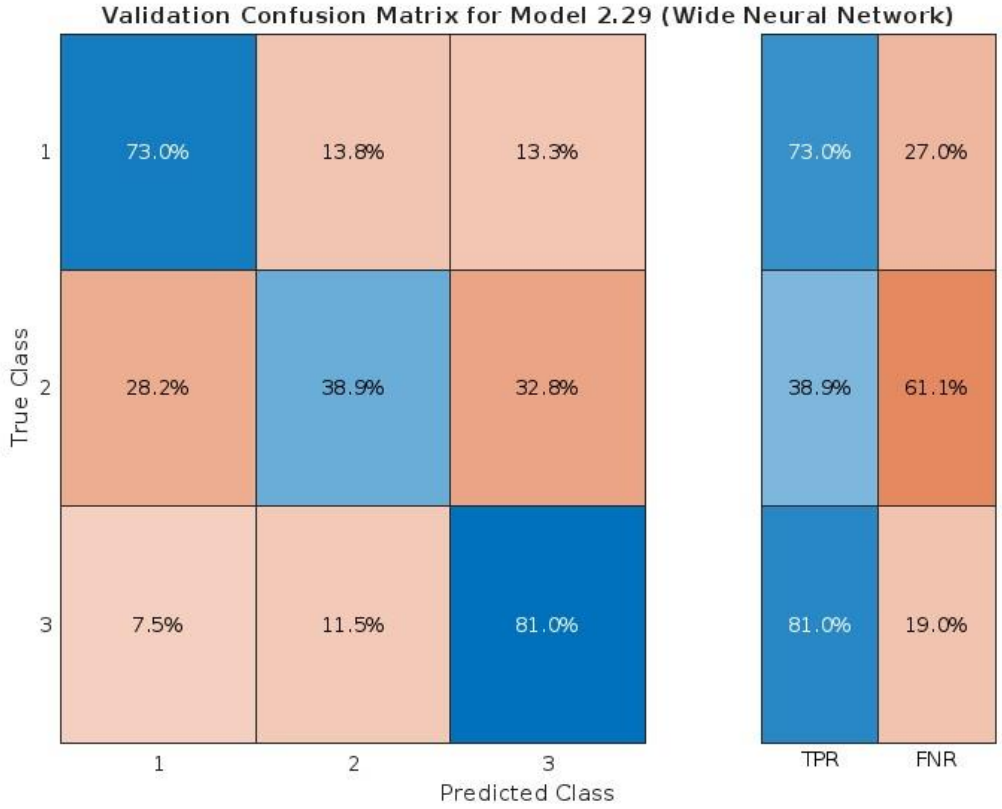


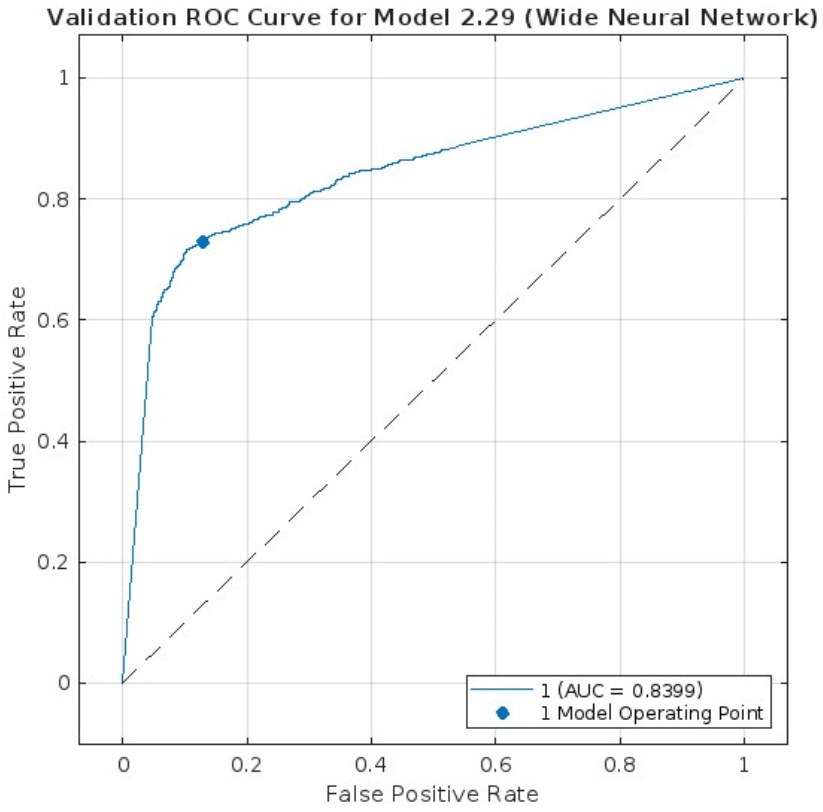
Testing graph



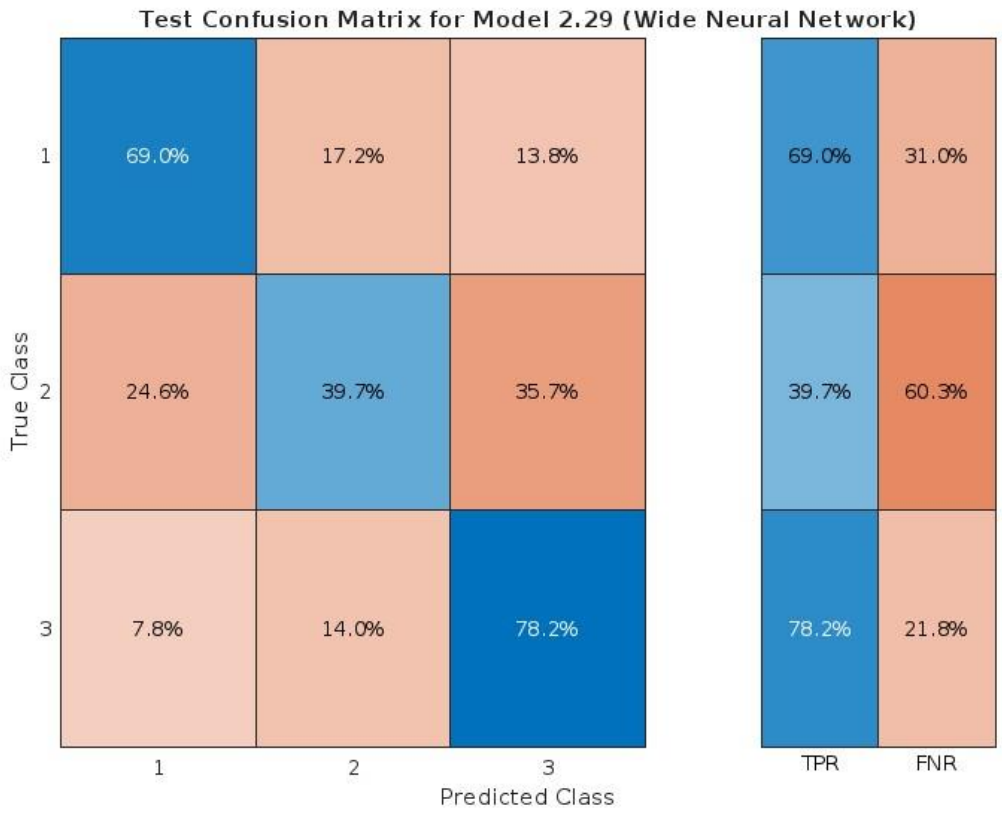


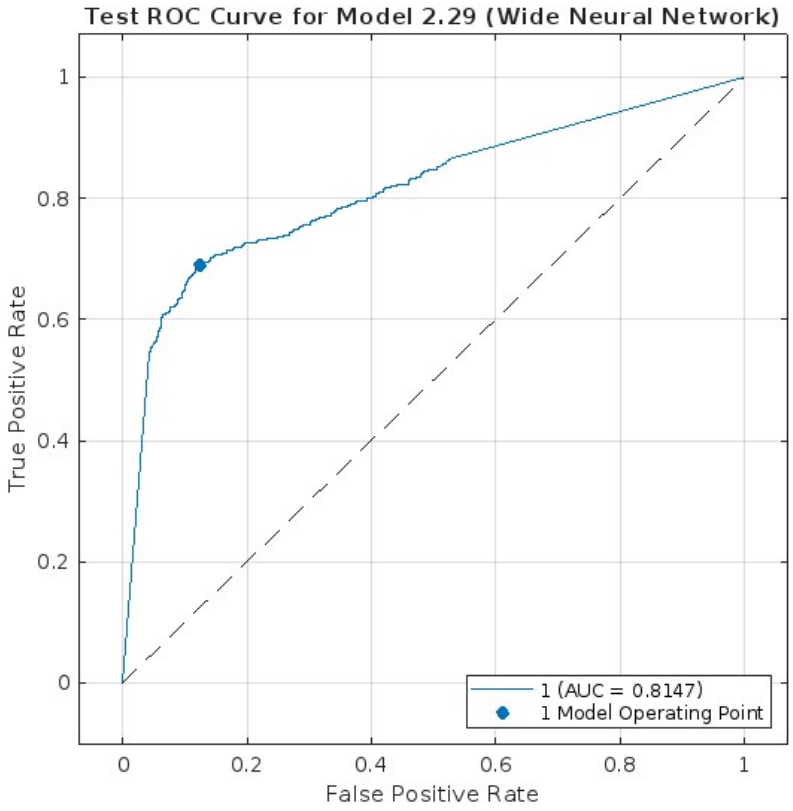
Wide Neural Network - Training graph



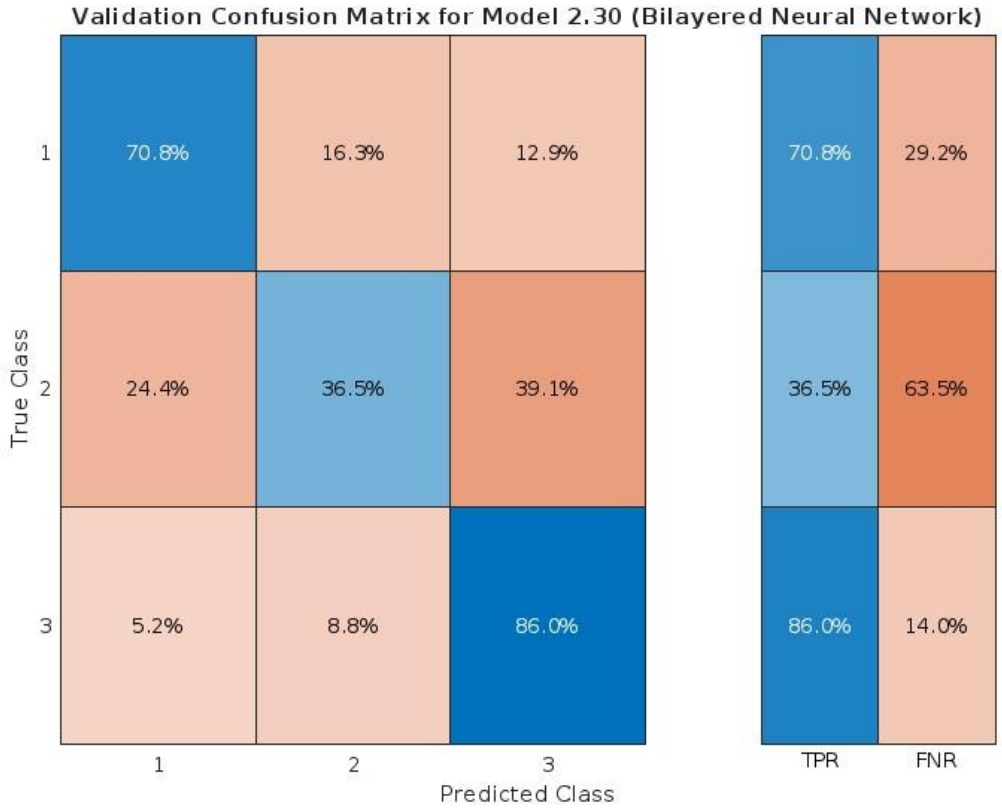


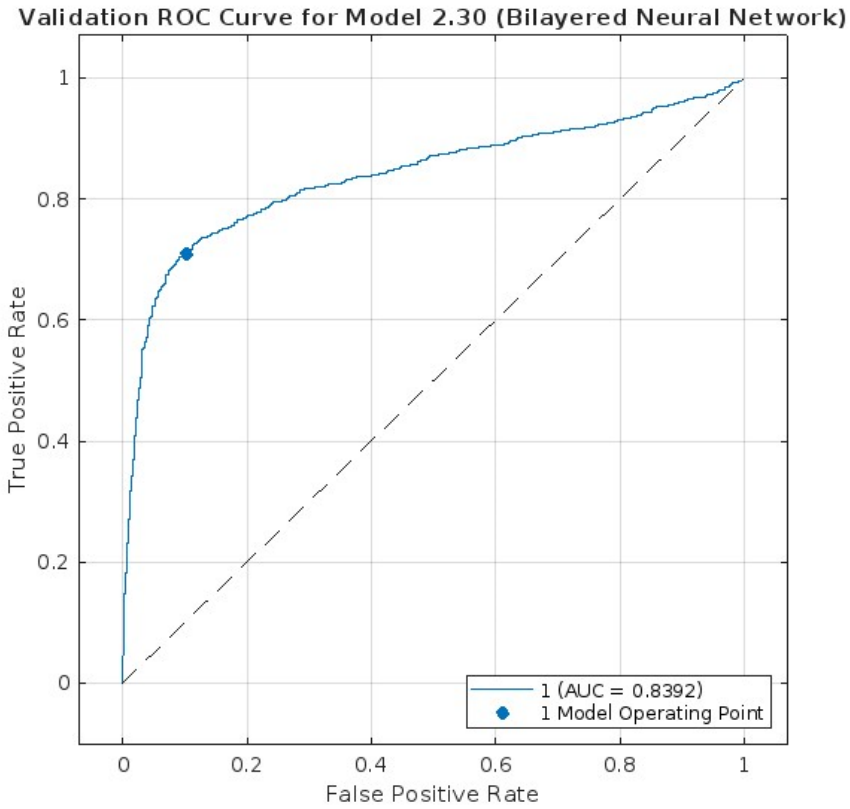
Testing graph



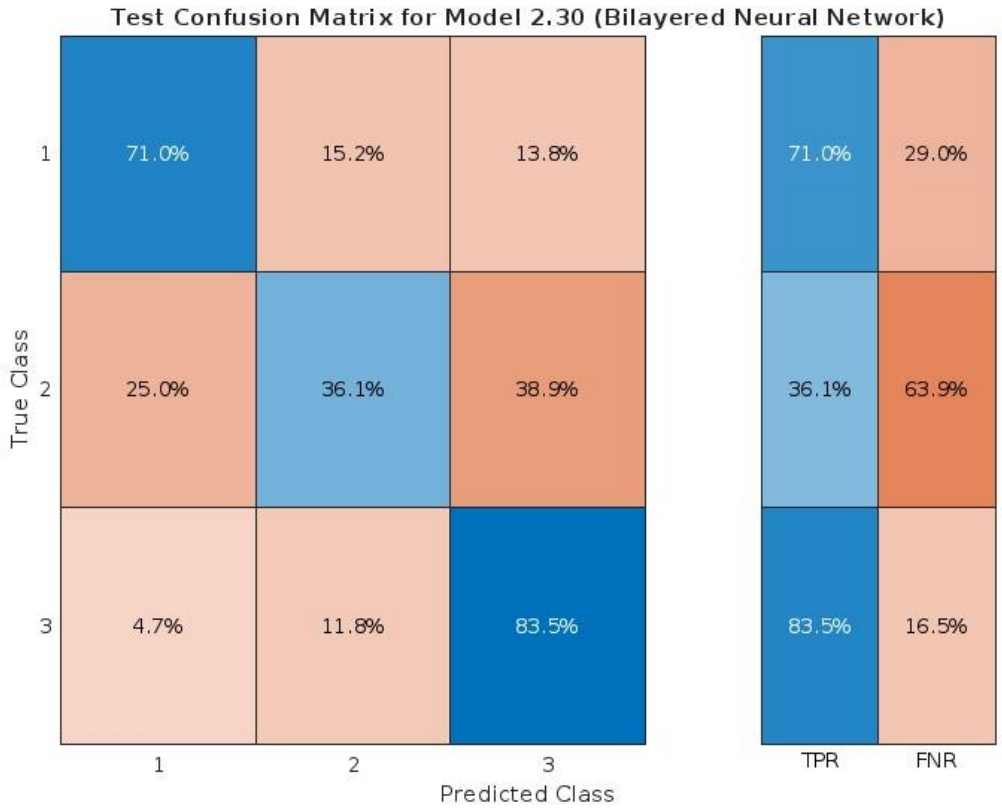


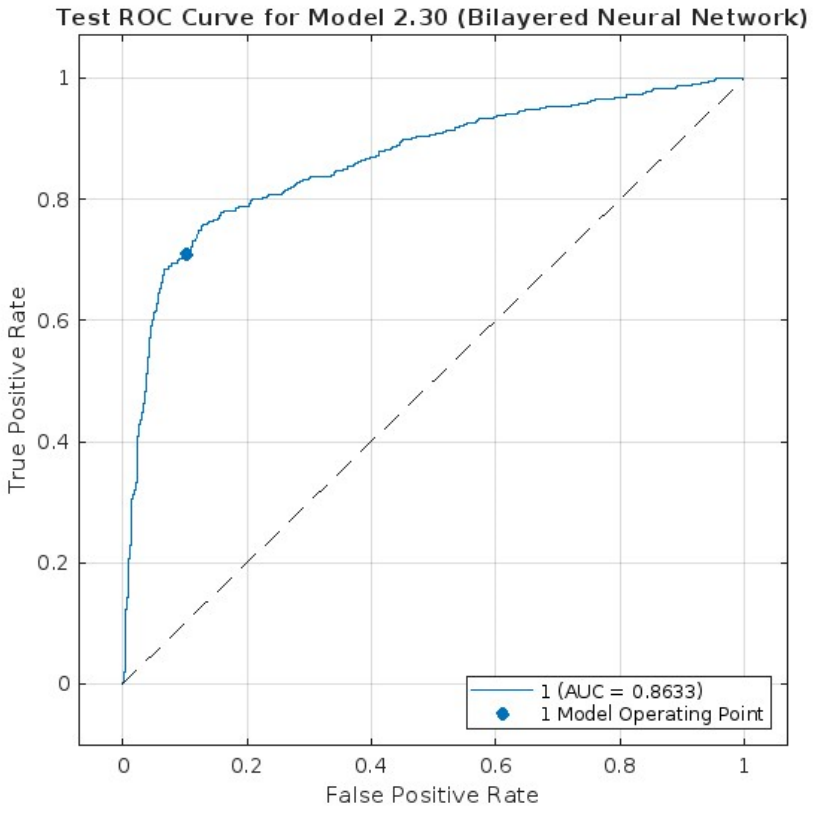
Bilayered neural Network - Training graph



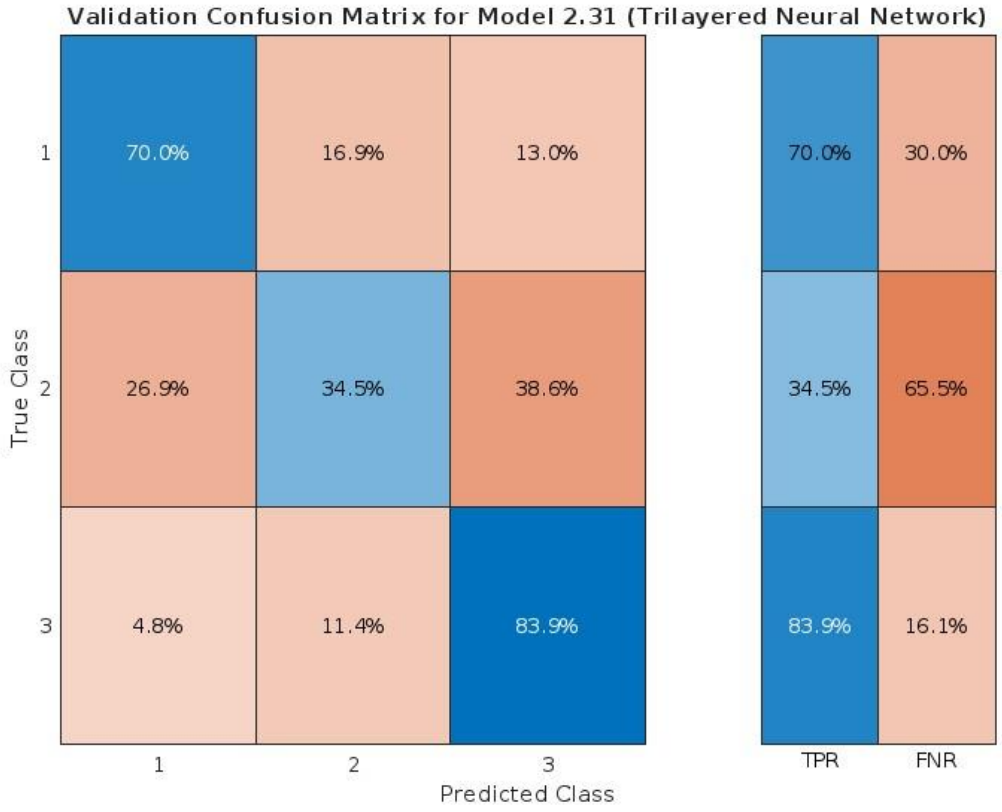


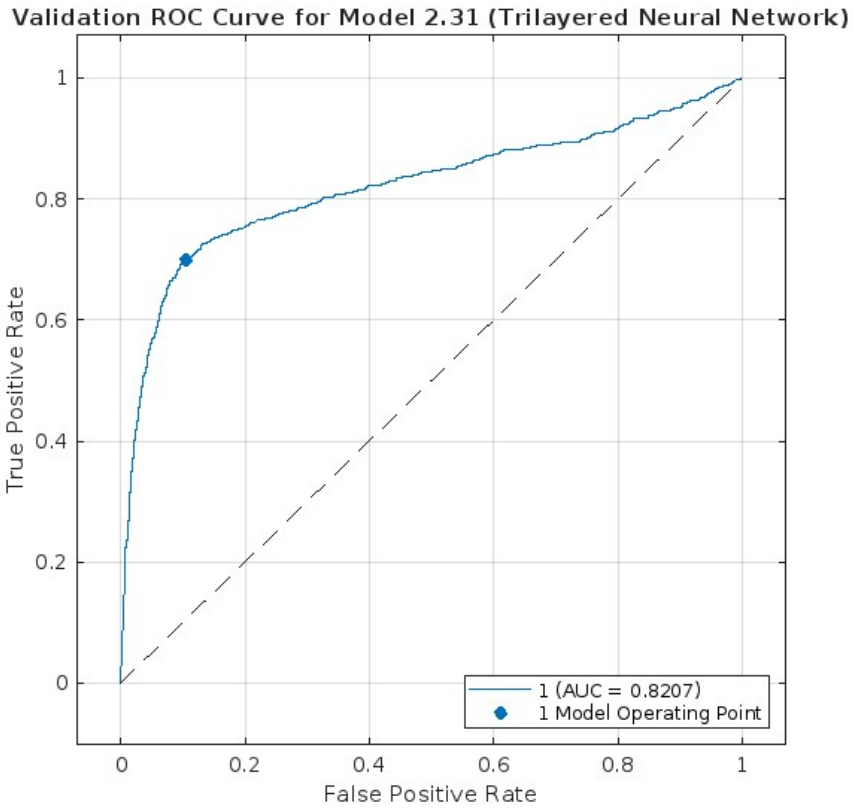
Testing graph



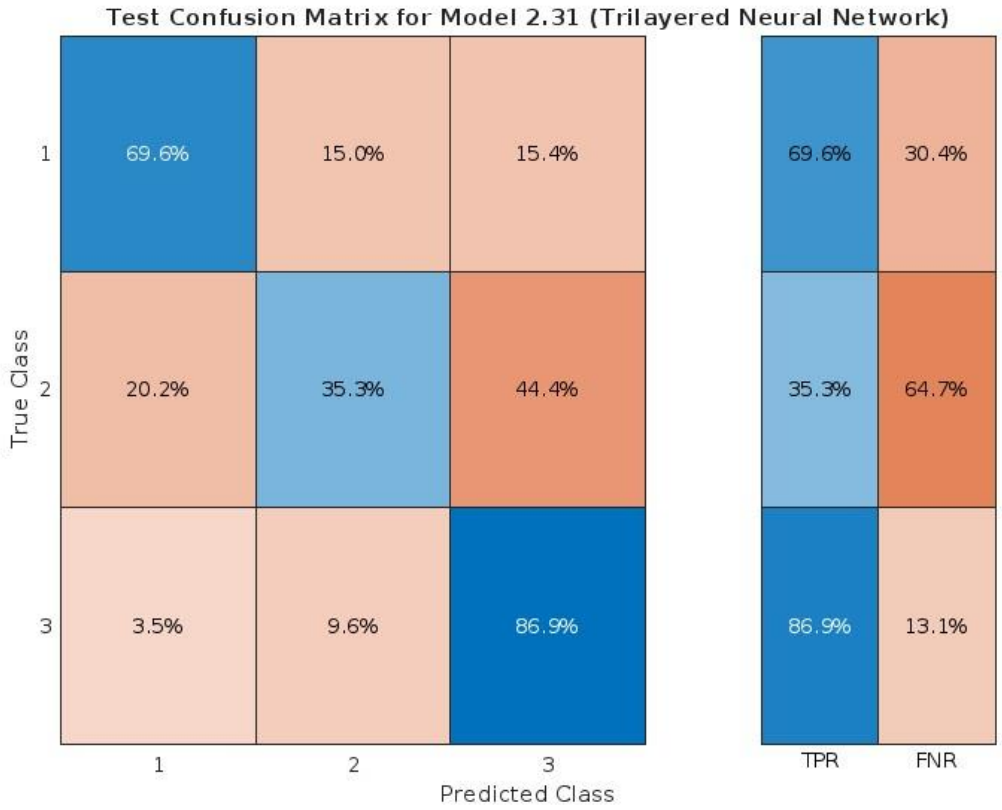


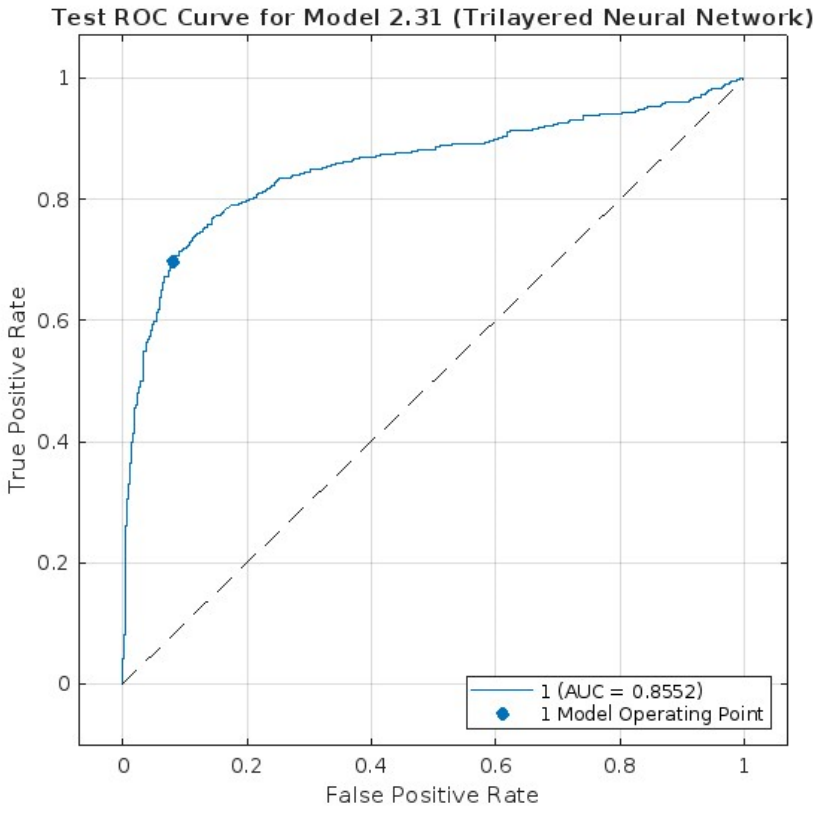
Tri layered Neural Network - Training graph



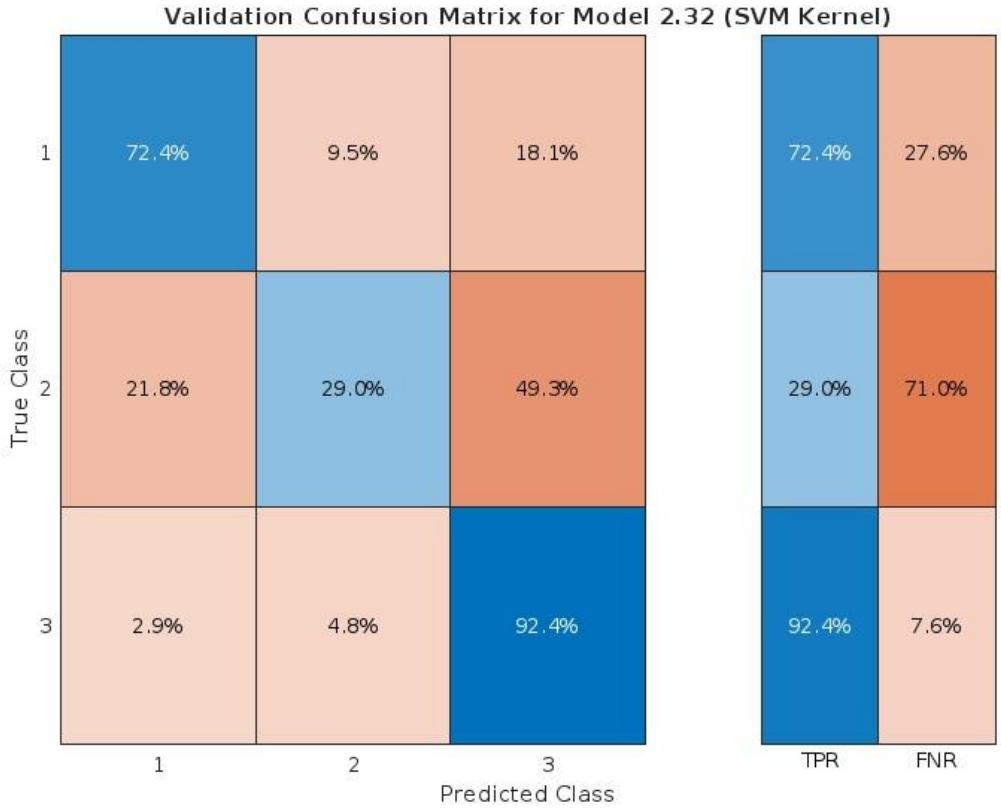


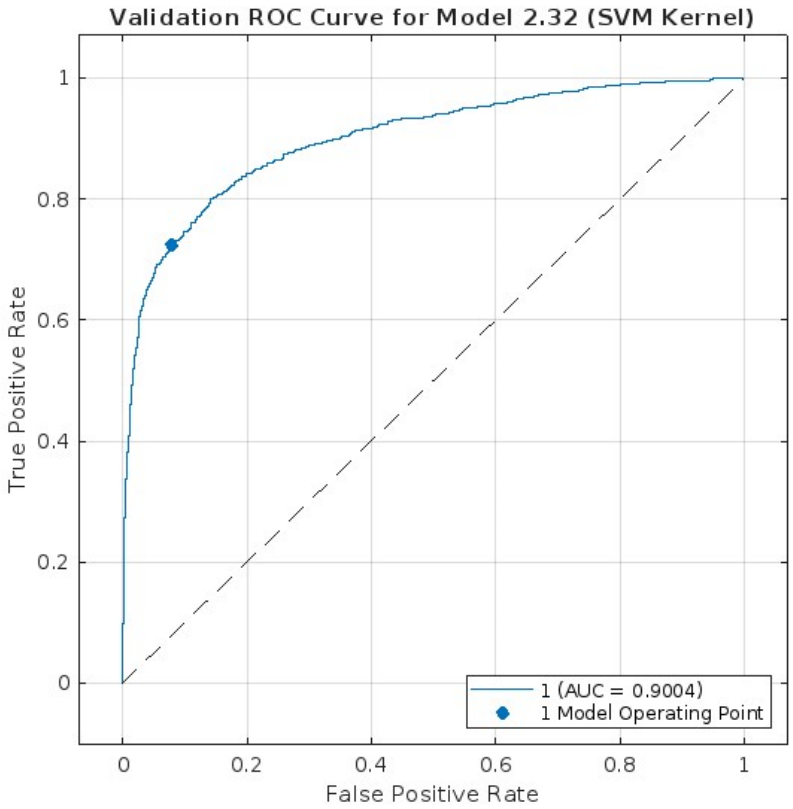
Testing graph



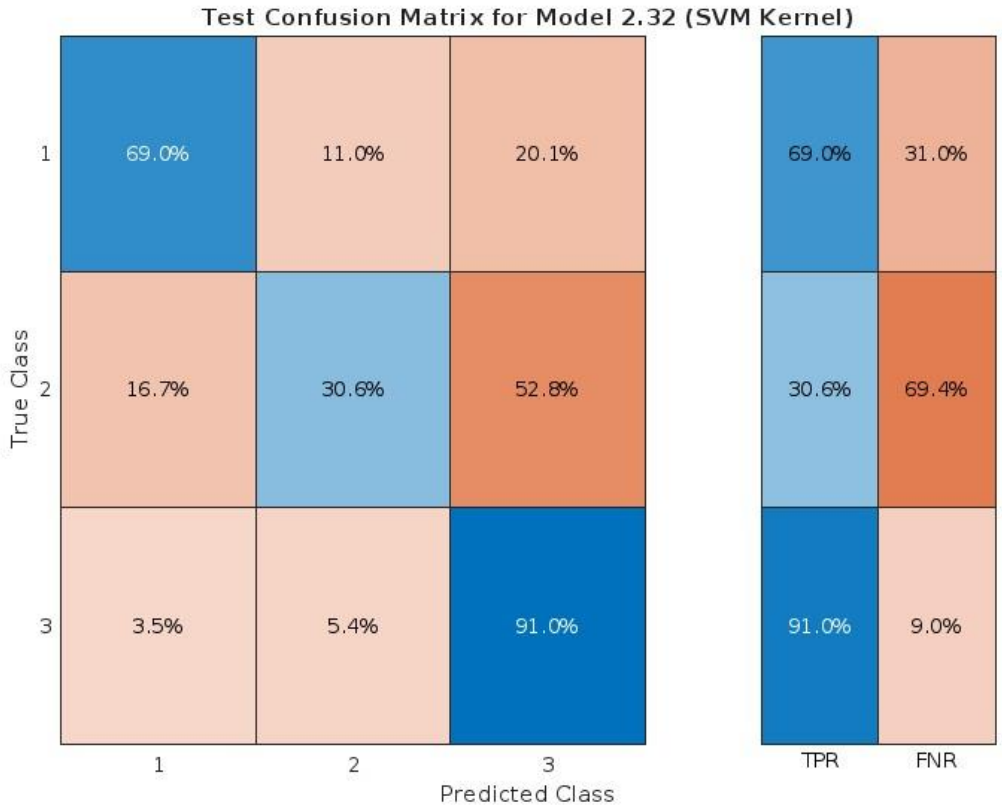


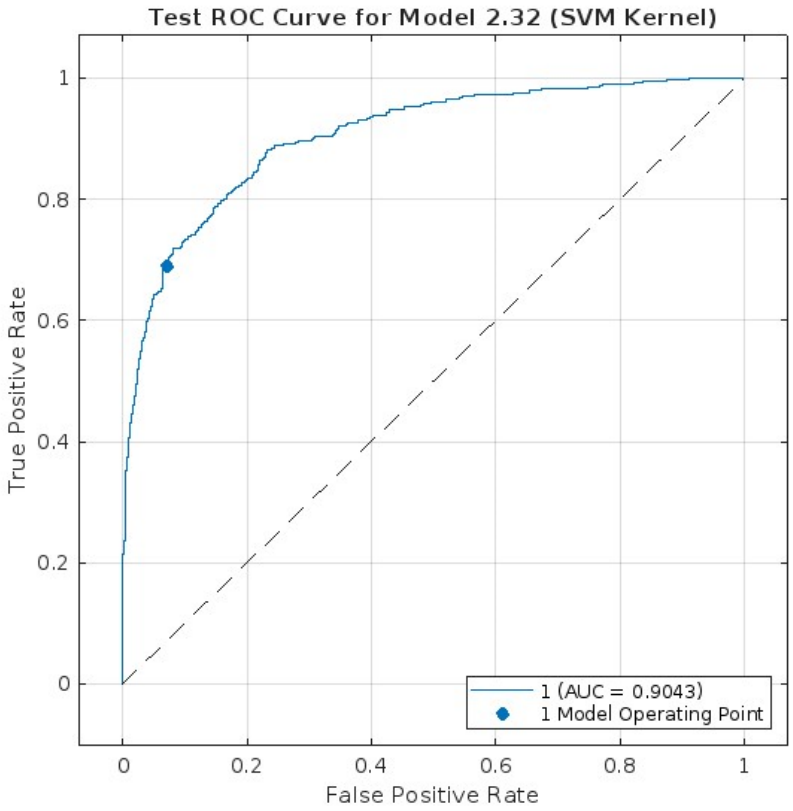
SVM Kernel


Validation Confusion Matrix for Model 2.32 (SVM Kernel)
True Class
Predicted Class
1
2
3
72.4%
9.5%
18.1%
21.8%
29.0%
49.3%
2.9%
4.8%
92.4%
72.4%
27.6%
29.0%
71.0%
92.4%
7.6%
TPR
FNR

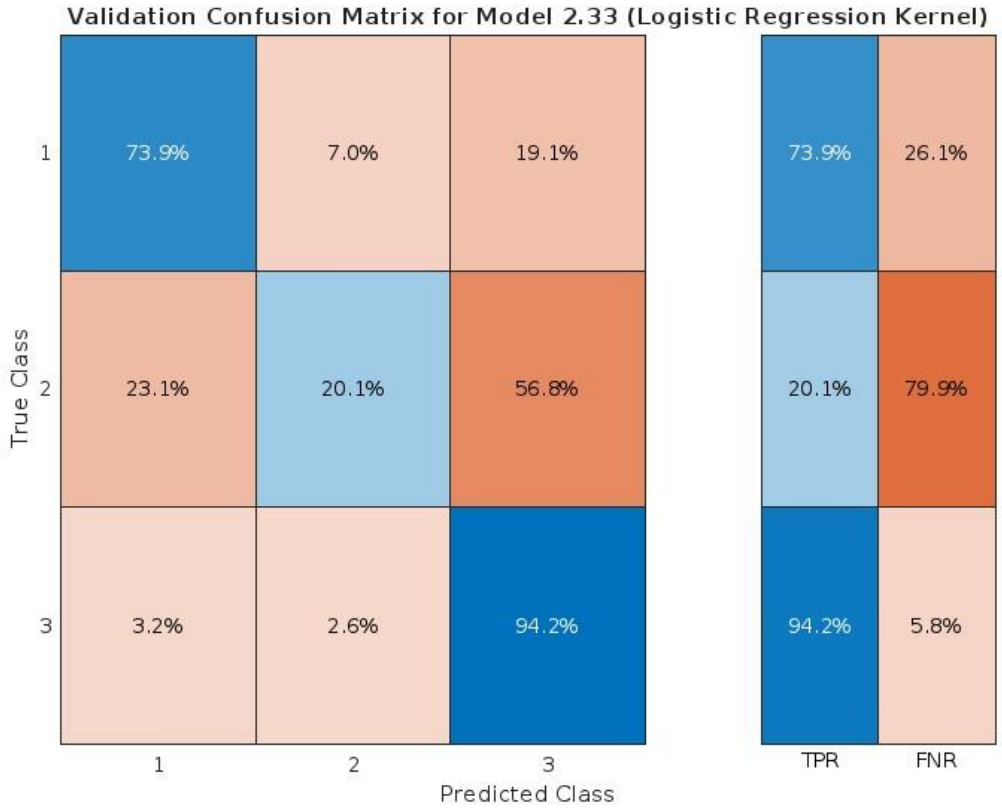

Validation ROC Curve for Model 2.32 (SVM Kernel)
True Positive Rate
False Positive Rate
0
0.2
0.4
0.6
0.8
1
1 (AUC = 0.9004)
1 Model Operating Point

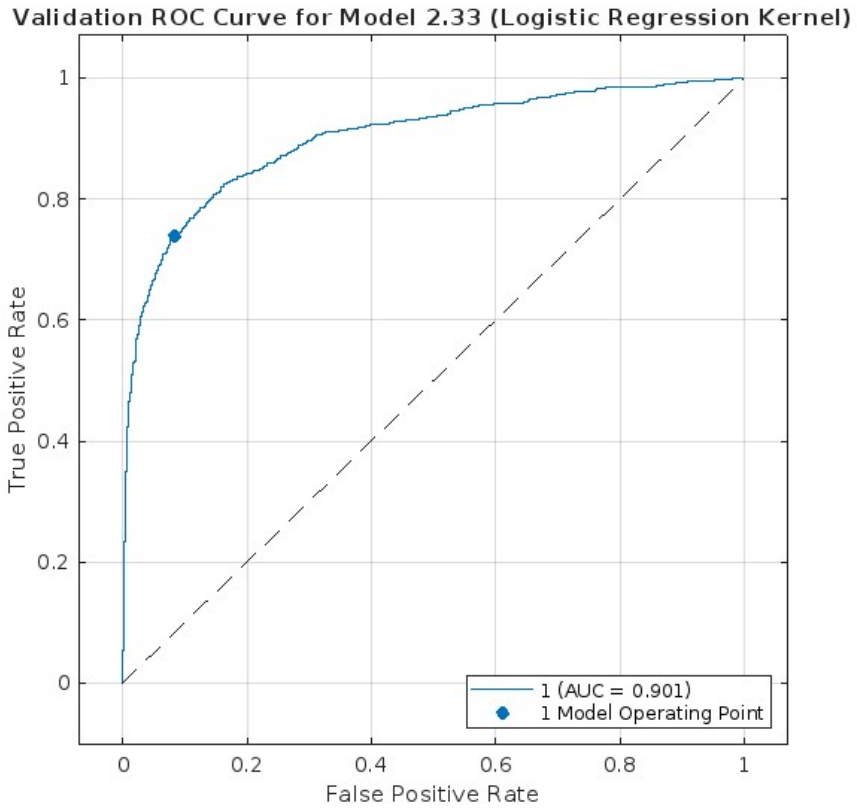
Testing graph



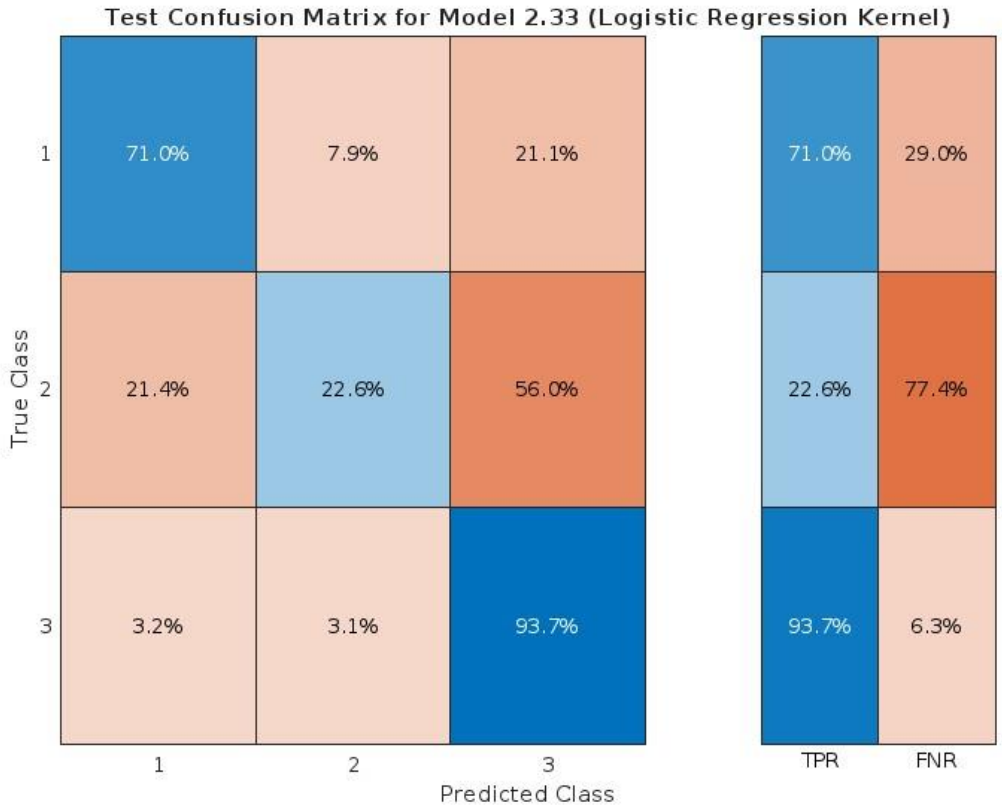


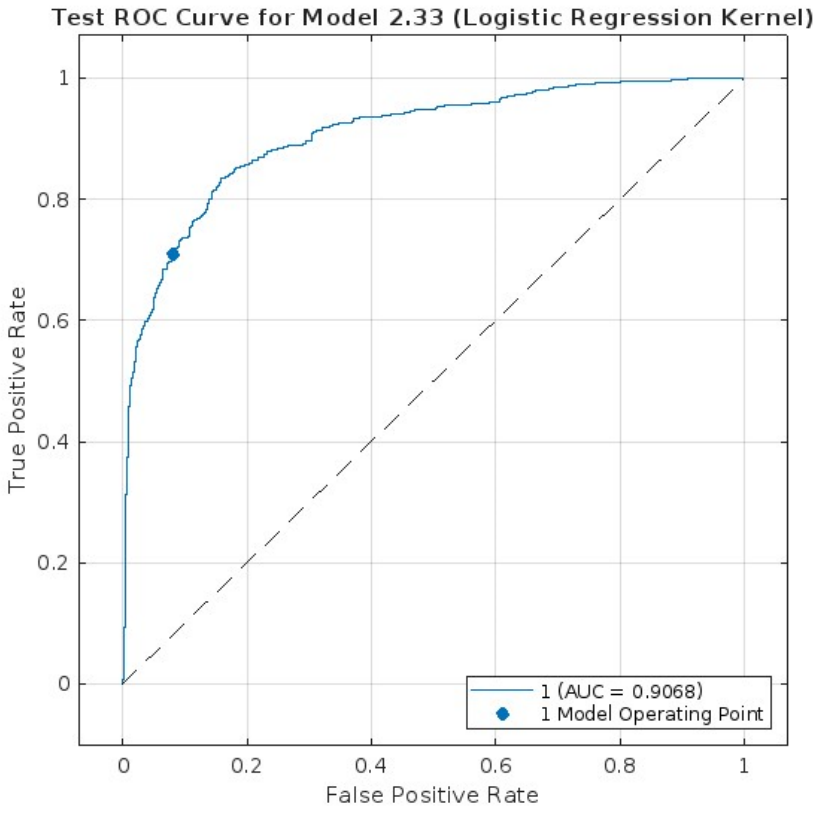
Logistics Regression Kernel





Testing graph



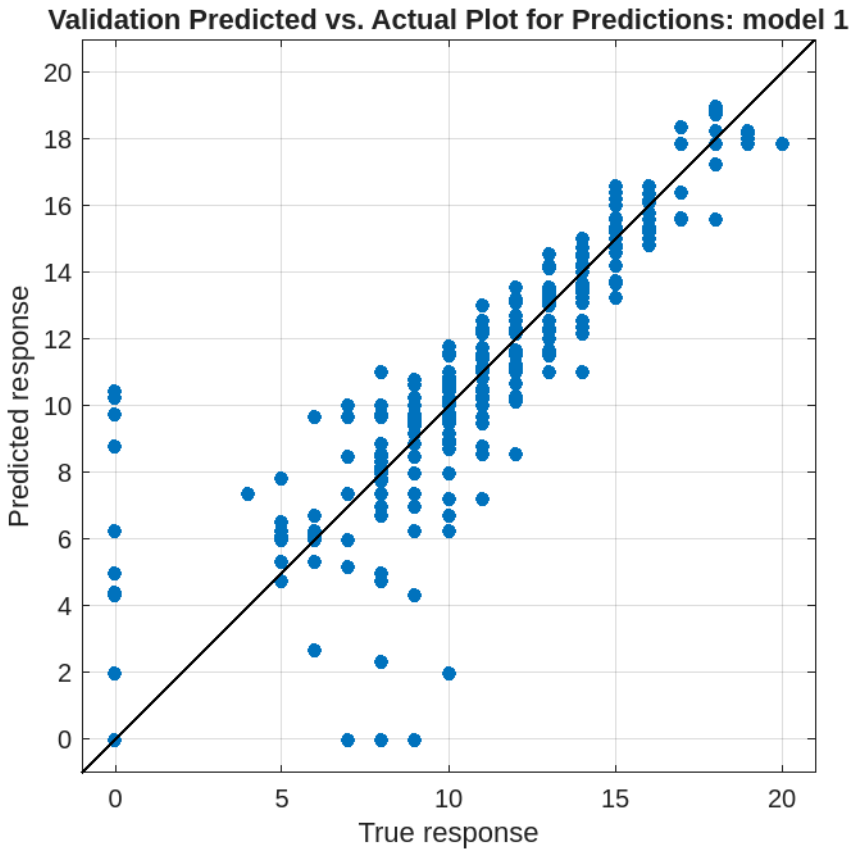


Regression Learner

I choose a regression learner dataset for predicting student performance because it allows me to explore how numerical target values, like final grades (G3), can be predicted using various features, such as demographics, social factors, and prior academic performance (G1 and G2). Regression tasks are ideal for understanding the relationships between continuous variables, such as how mid-year grades or social factors influence final grades. This dataset provides a great opportunity for me to practice feature engineering, address multicollinearity (e.g., the strong correlation between G3, G2, and G1), and apply regression algorithms while gaining valuable insights into modeling real-world academic outcomes.

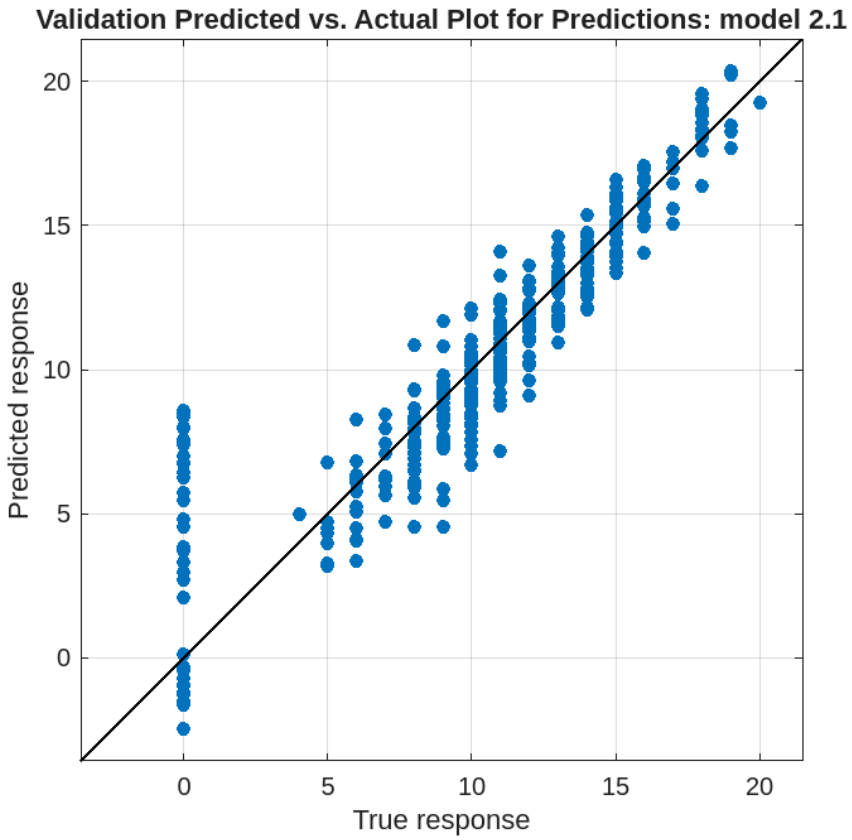
As it relates to preprocessing and model training, the dataset was already in CSV format, so I began by importing the data into MATLAB. Once the data was imported, I proceeded to import the selection of regression algorithms available in MATLAB, which allowed me to train the dataset using ALL algorithms. This gave me the flexibility to experiment with different algorithms, such as linear regression, medium tree, and quadratic SVM. After training the models, I analyzed the performance by examining the results table, which provided detailed metrics like RMSE, R-squared, and other performance indicators. To visualize the model's accuracy, I created a predicted vs. actual plot, which helped me understand how well the model's predictions aligned with the true values. This process allowed me to compare the performance of each algorithm and determine the best one for the regression task at hand. My analysis and conclusions are further explained in this document:

Fine Tree

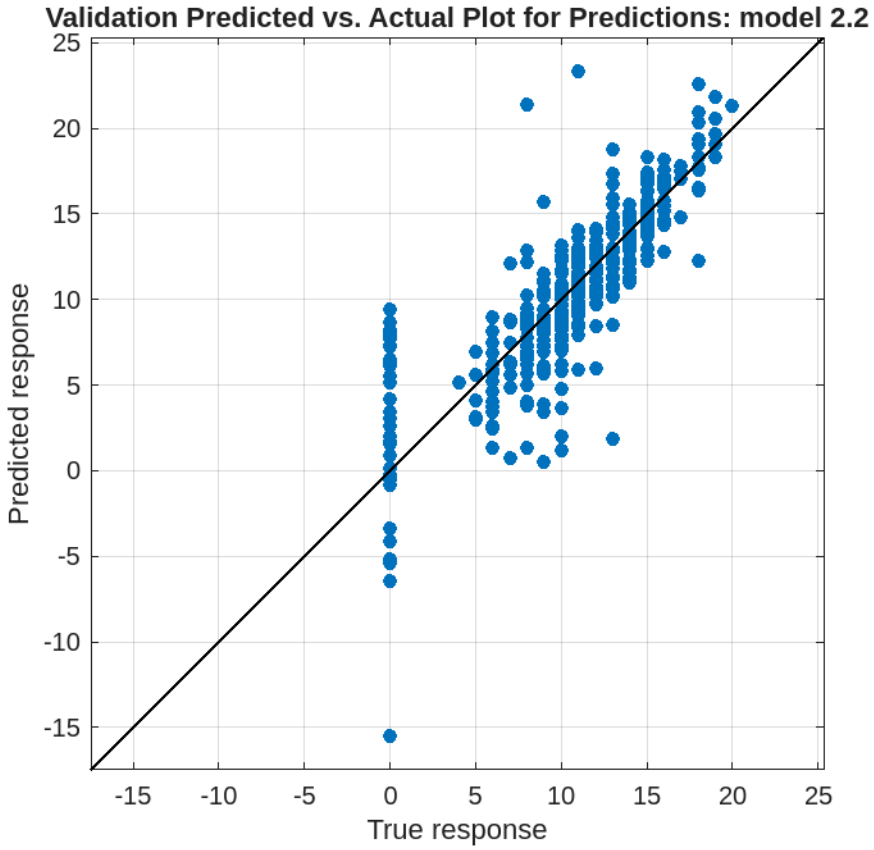


Description: This graph compares the predicted values from a regression model to the actual values in the validation dataset. The diagonal line represents perfect predictions, where predicted and actual values match. Points close to the line indicate accurate predictions, while scattered points show prediction errors. The model performs reasonably well overall but has some inaccuracies, especially at extreme values.

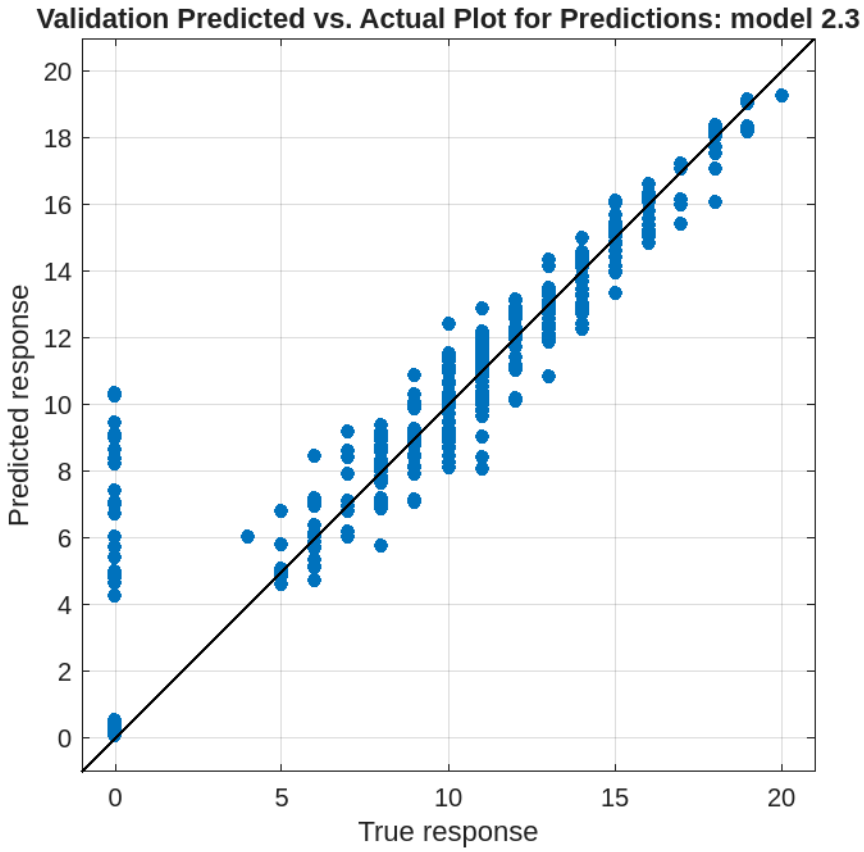
Linear Regression



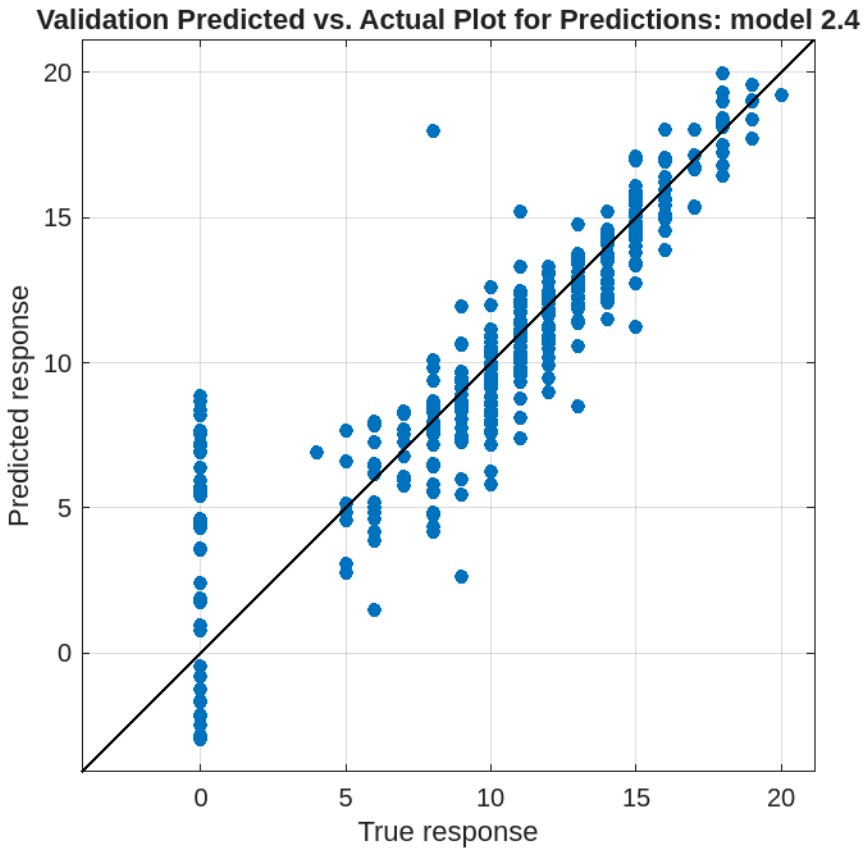
Interactions Linear Regression



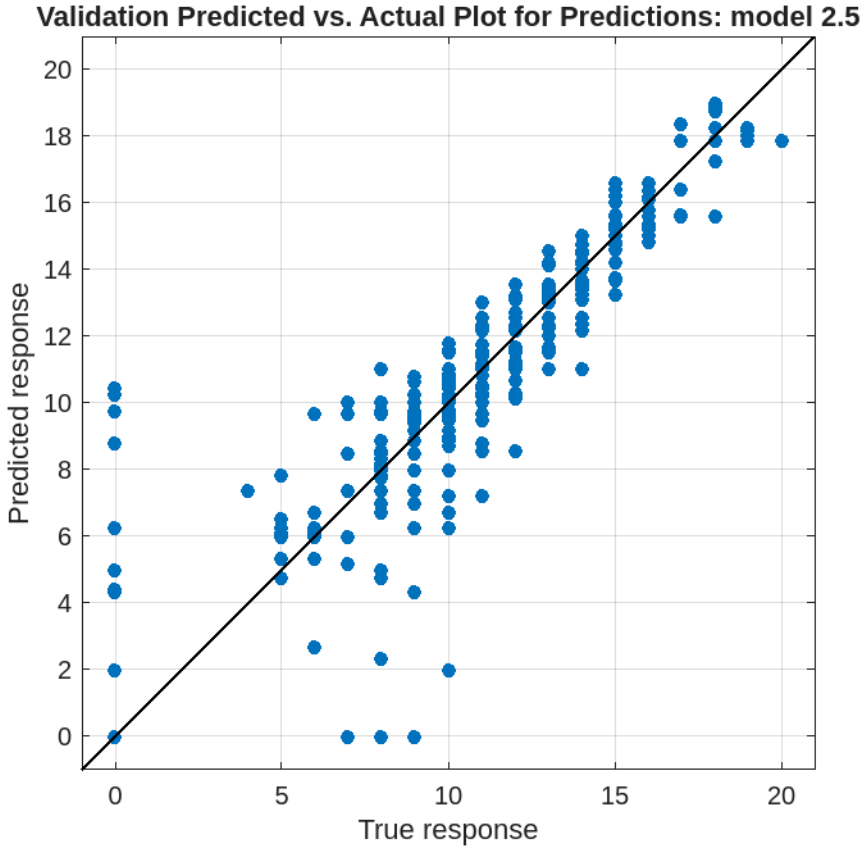
Robust Linear Regression



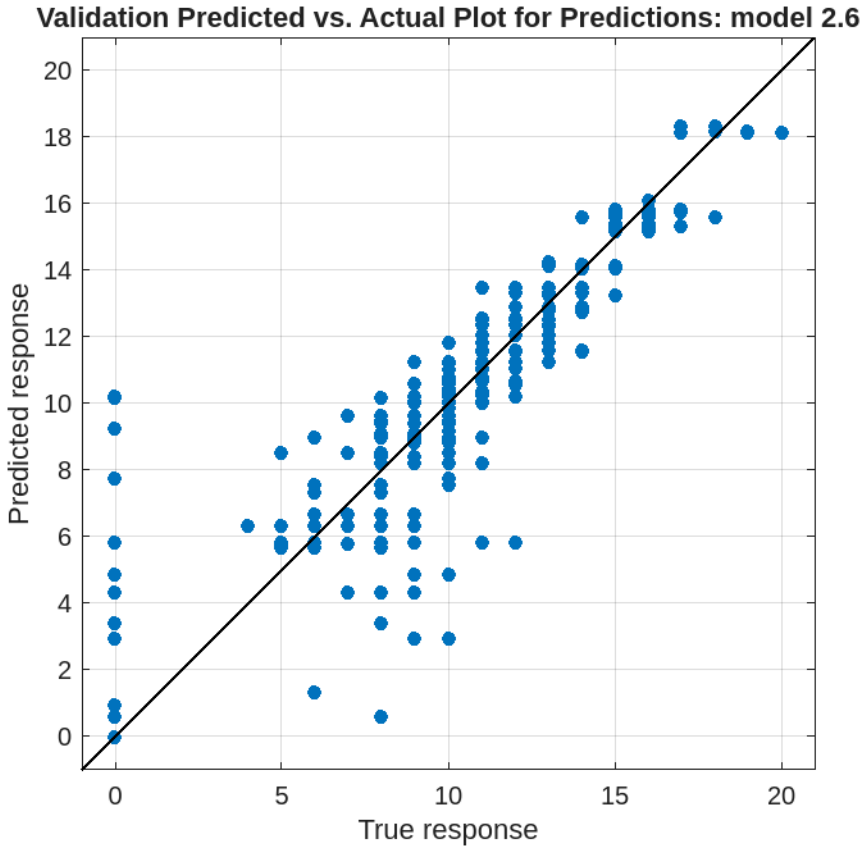
Stepwise linear regression



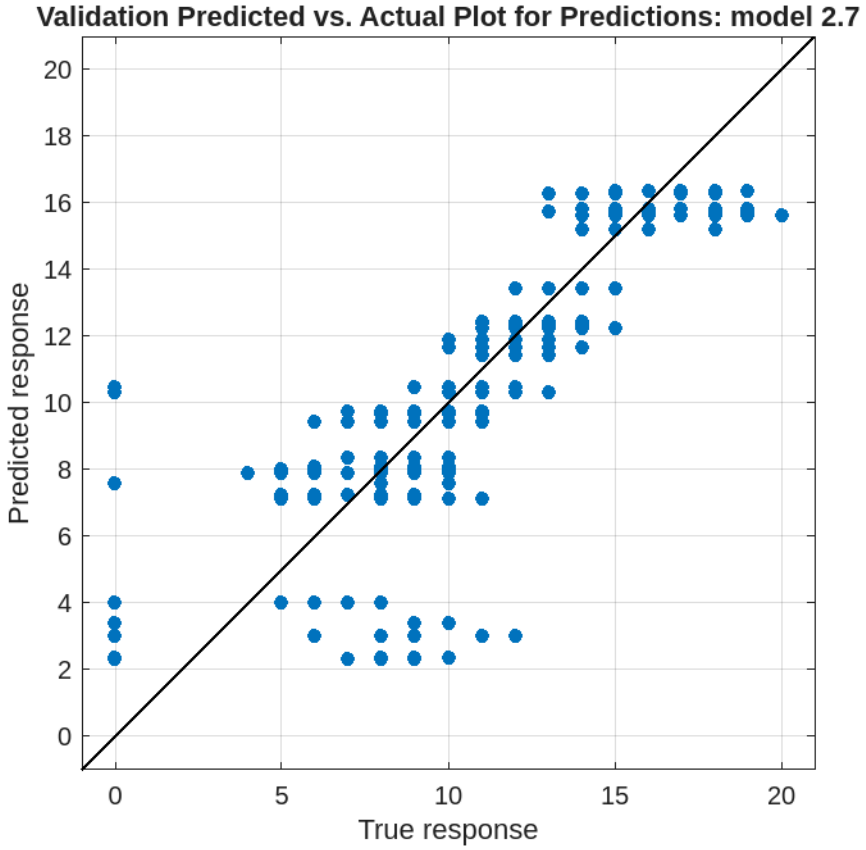
Fine tree Graph



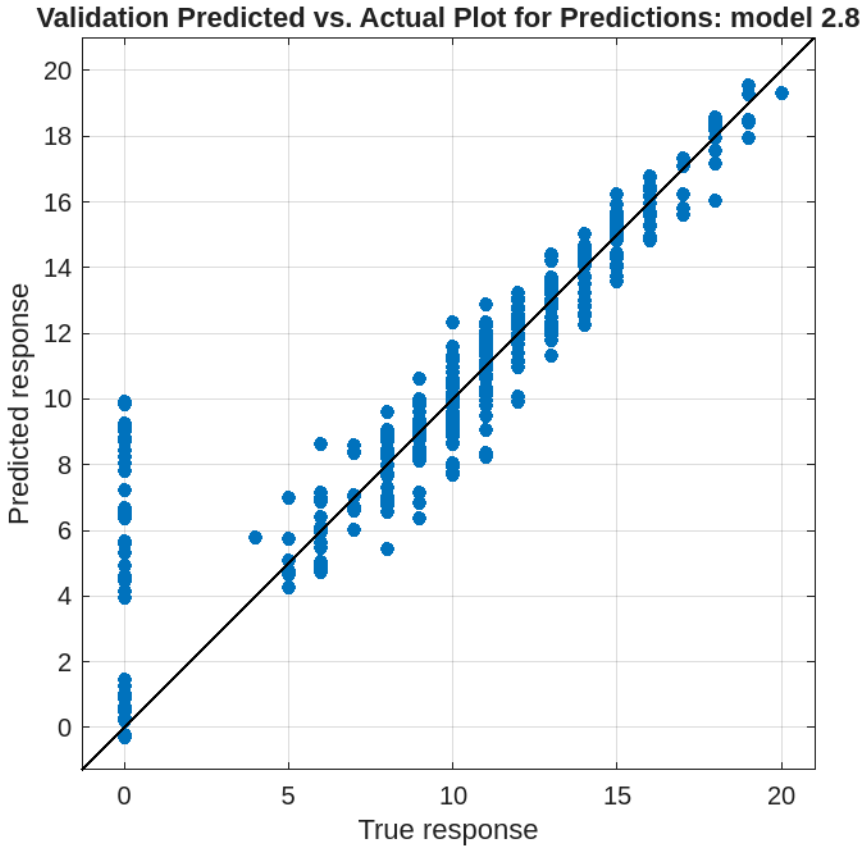
Medium Tree



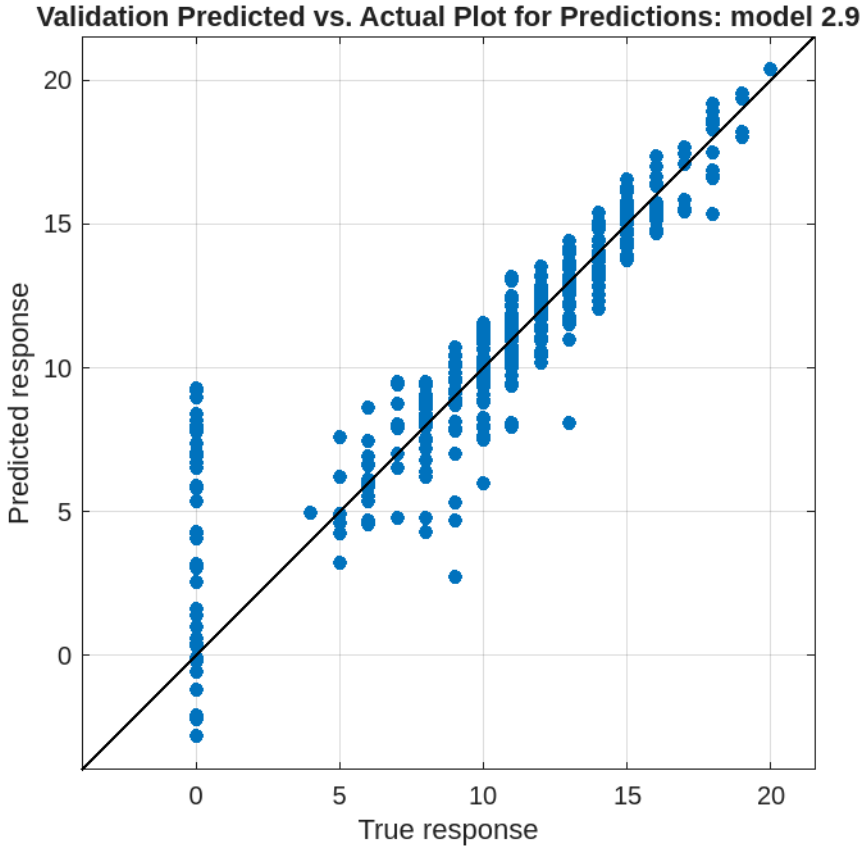
Coarse Tree



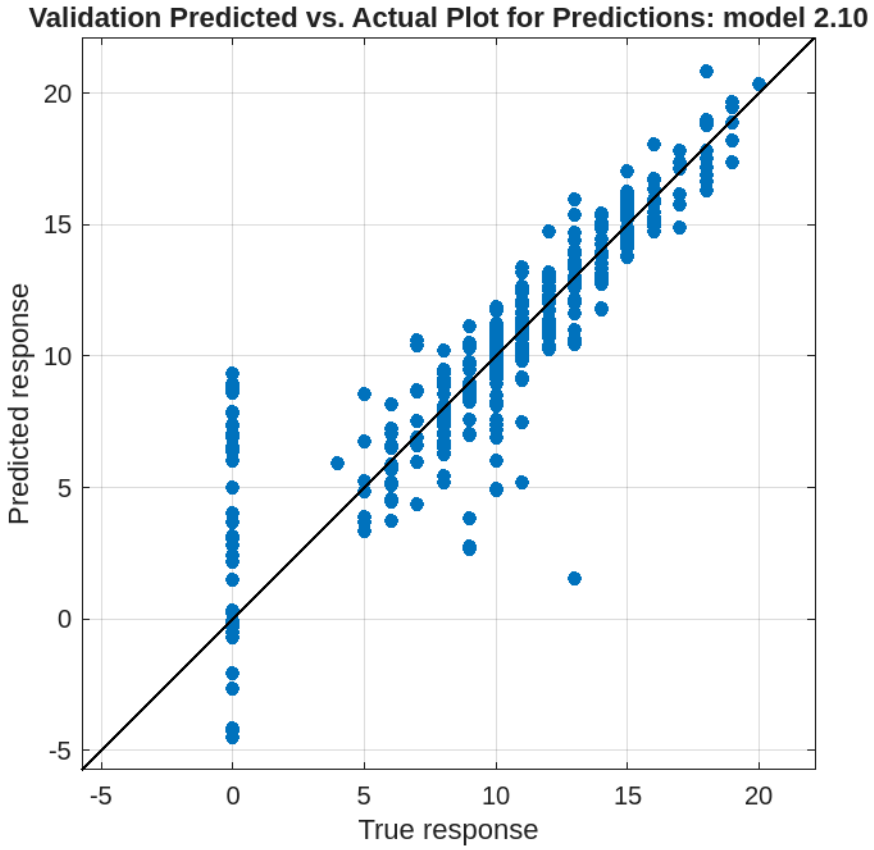
Linear SVM



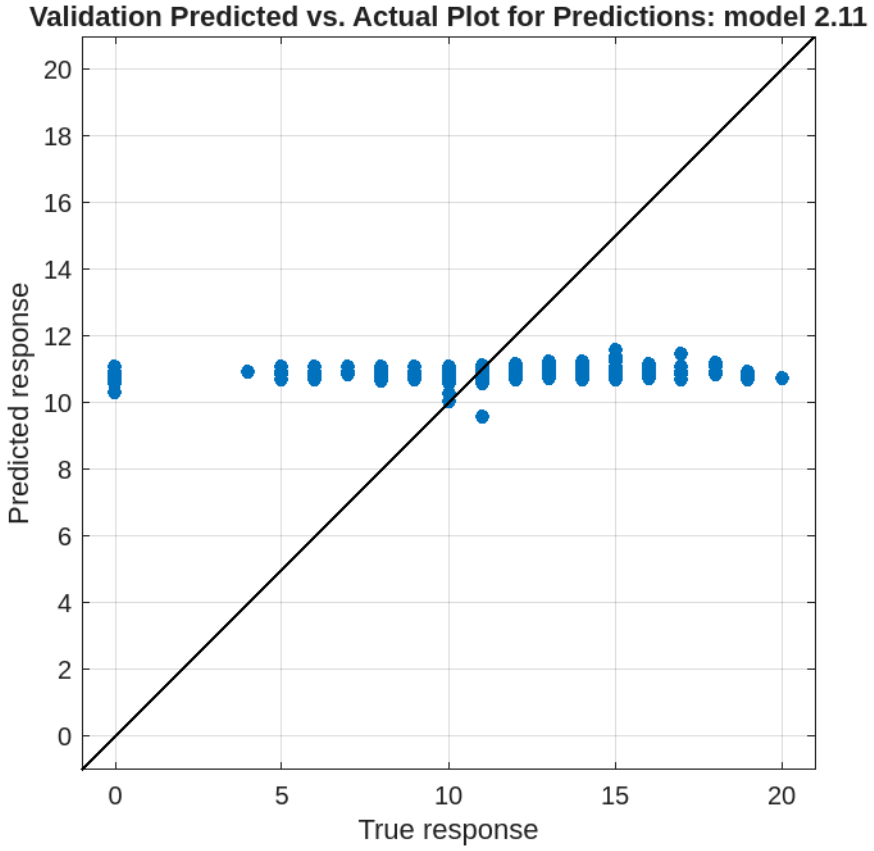
Quadratic SVM



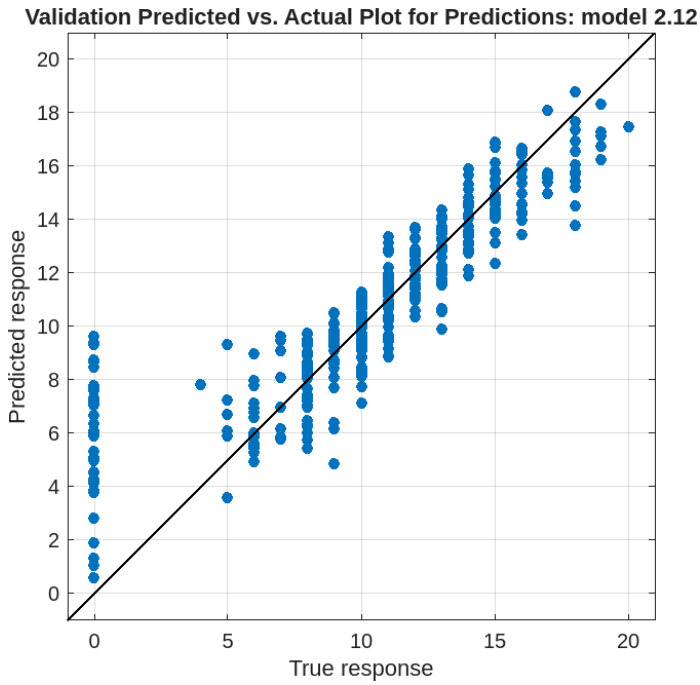
Cubic SVM



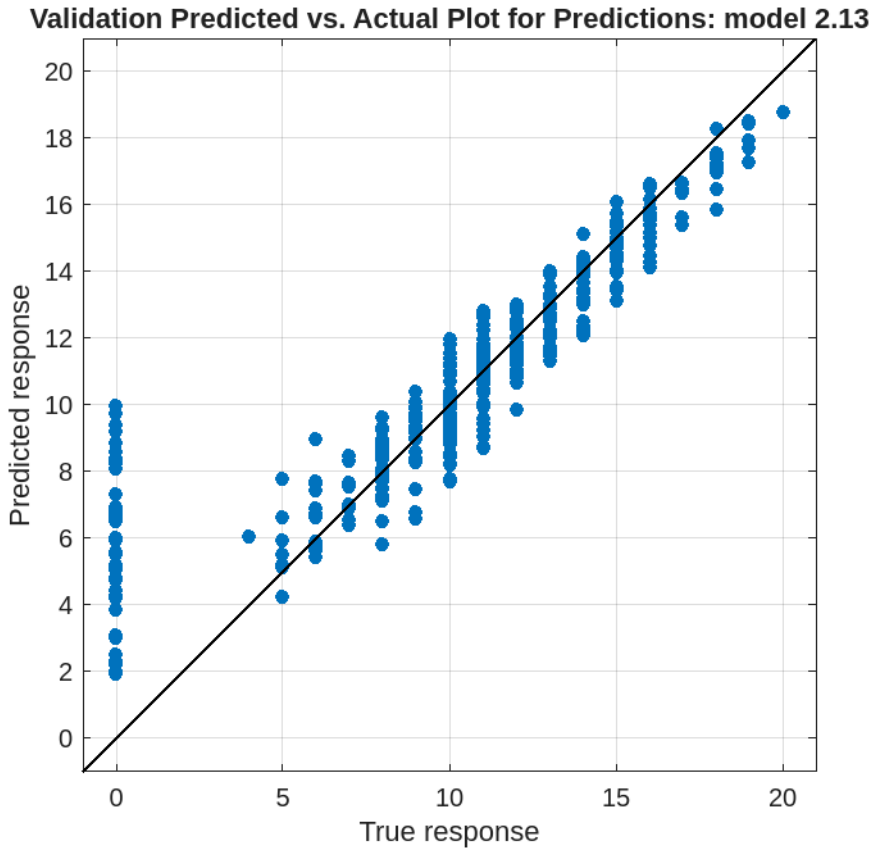
Fine Gaussian SVM



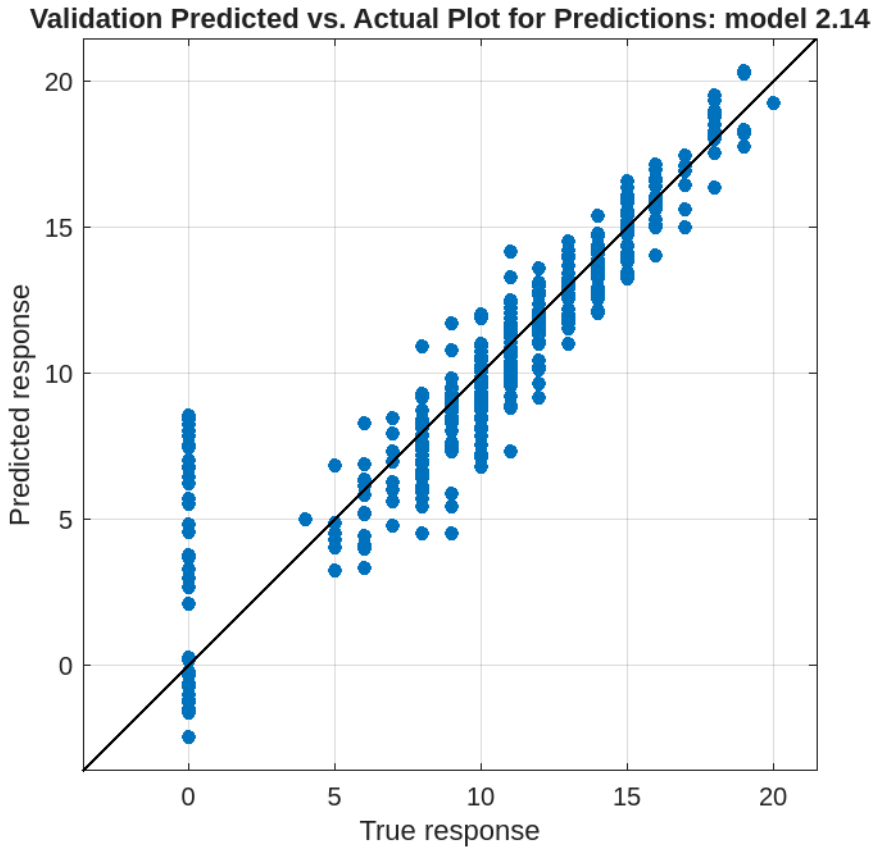
Medium gaussian SVM



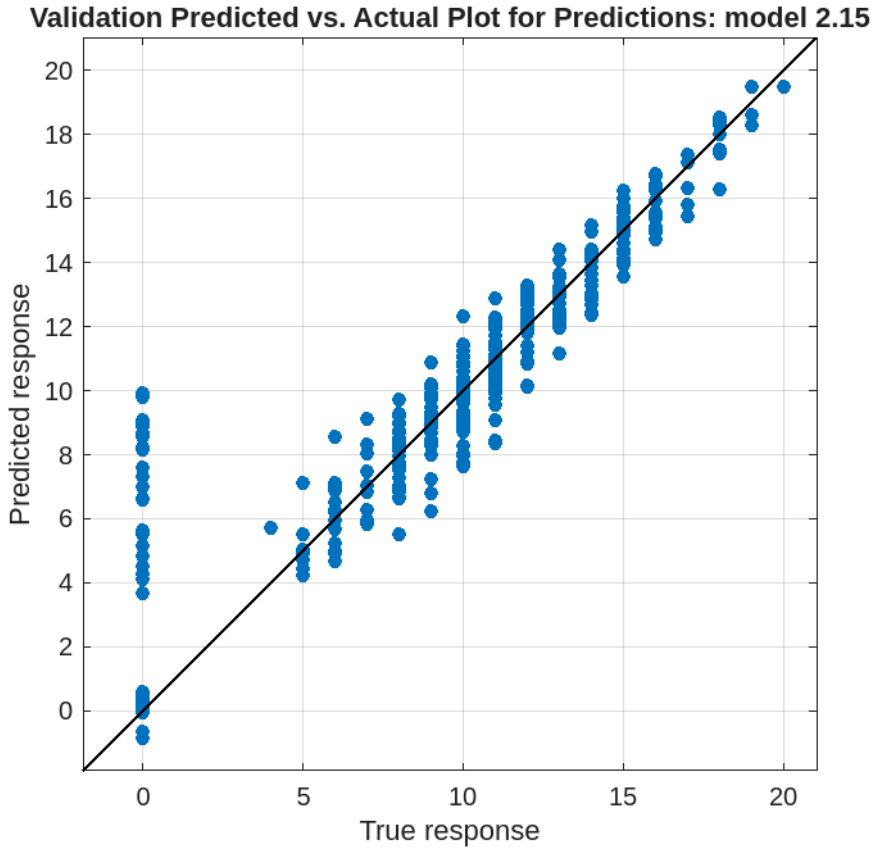
Coarse gaussian SVM



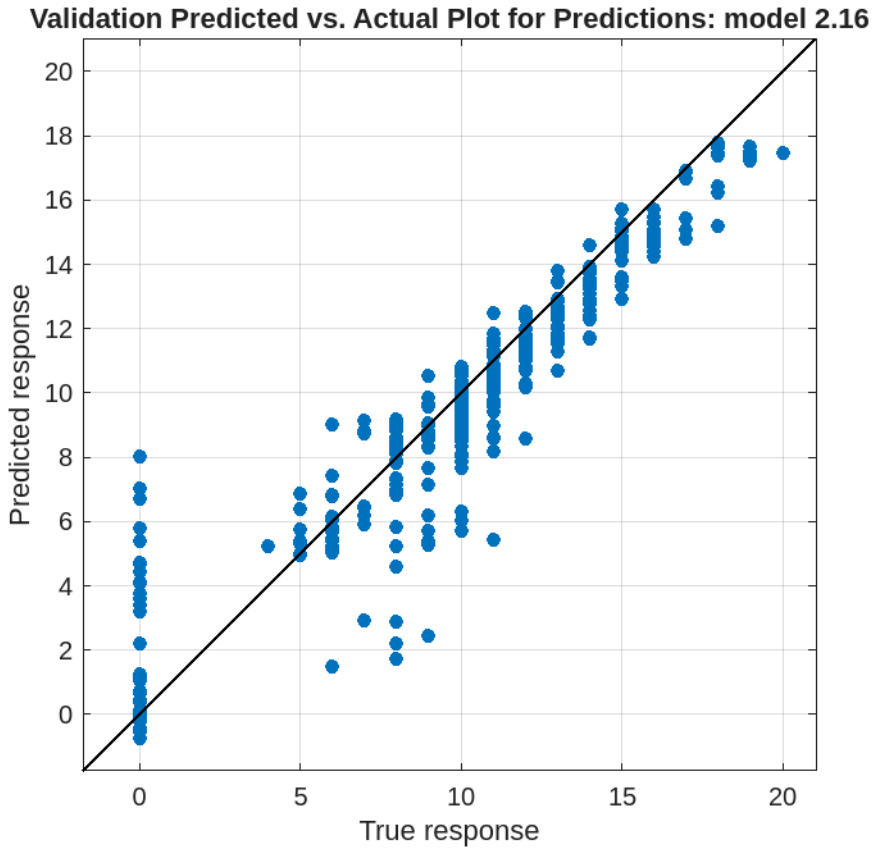
Efficient linear least squares



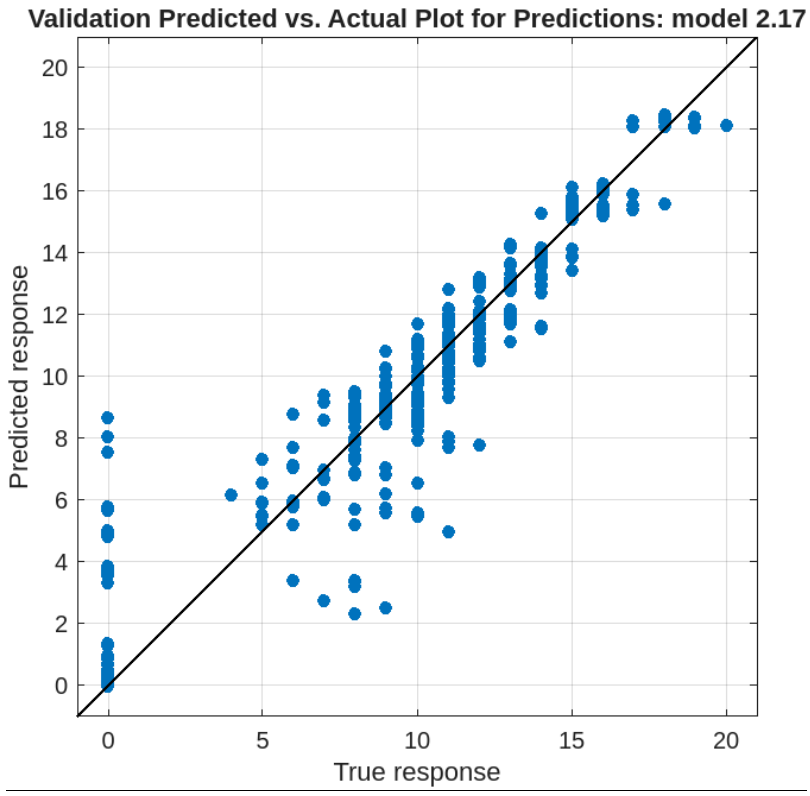
Efficient linear SVM



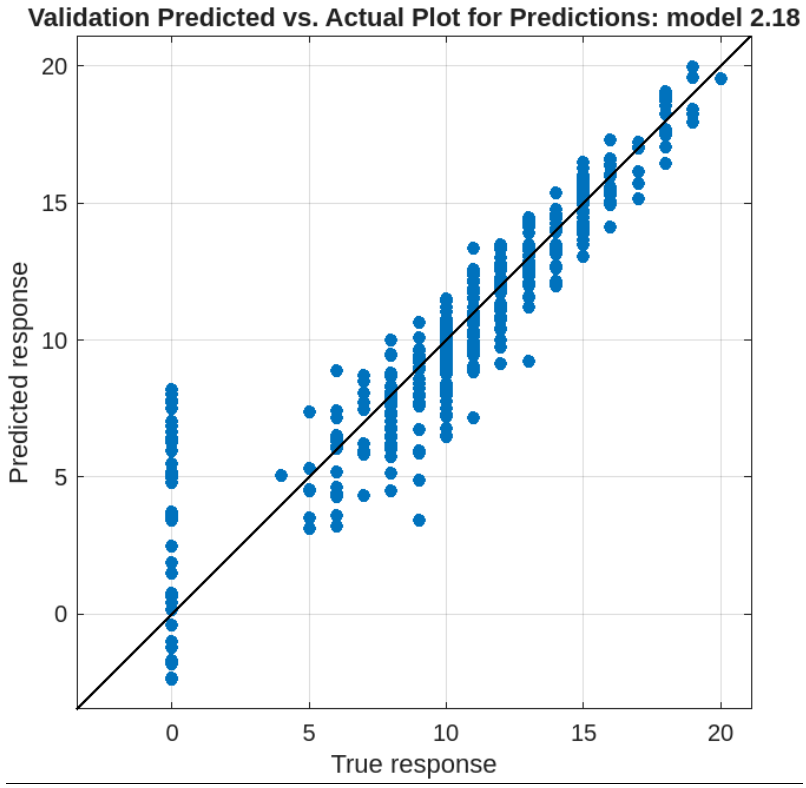
Boosted trees



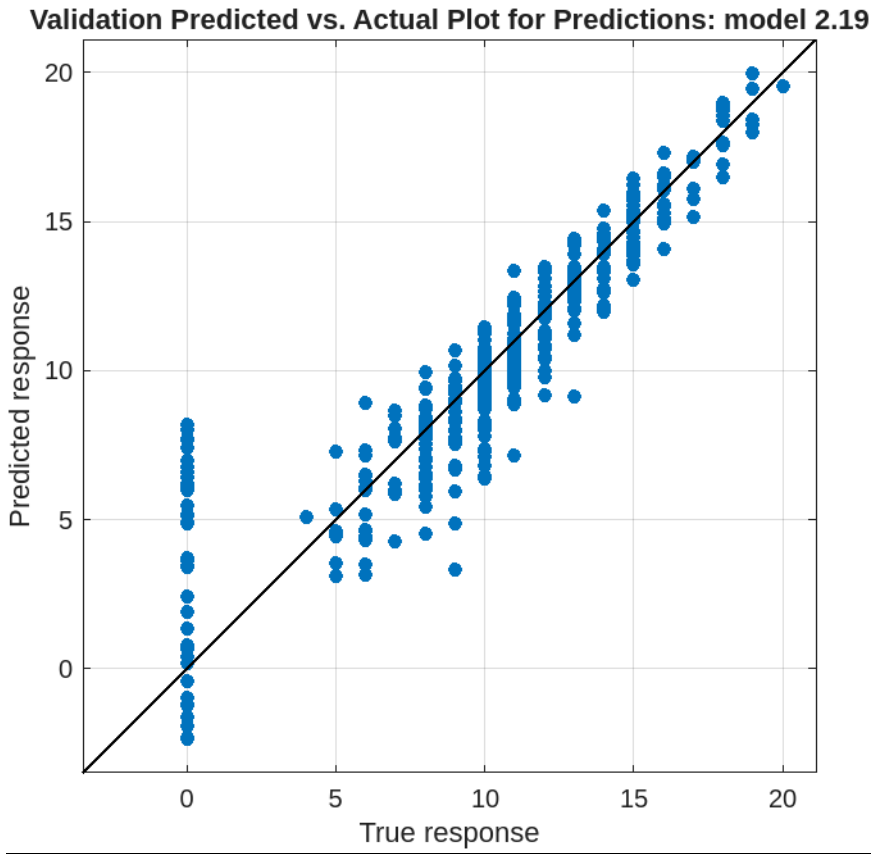
Bagged trees



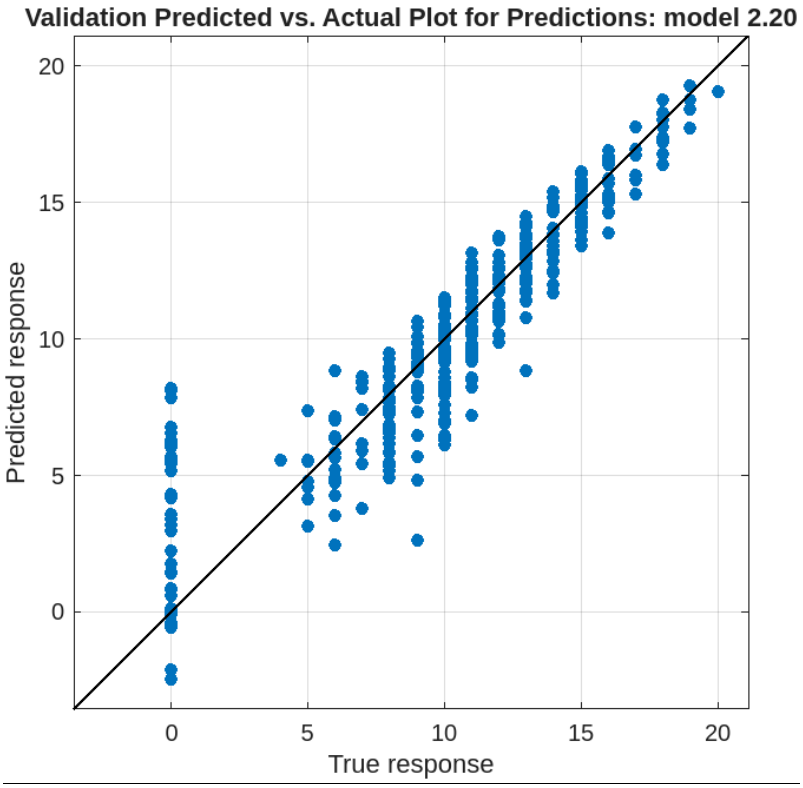
Squared exponential GPR



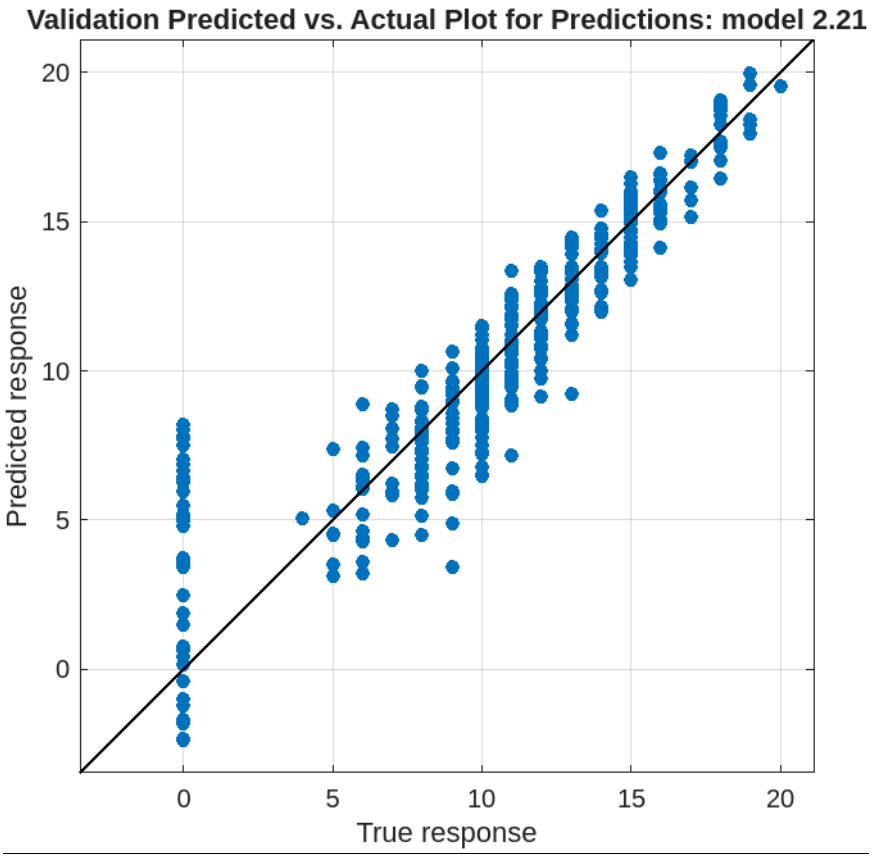
Matern 5/2 GPR



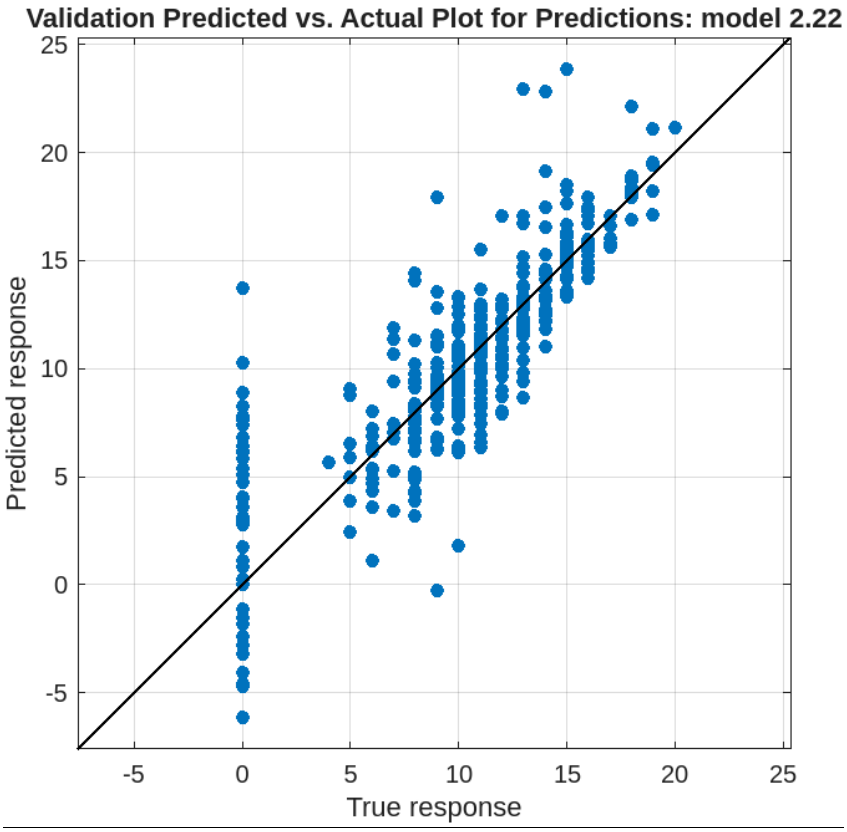
Exponential GPR



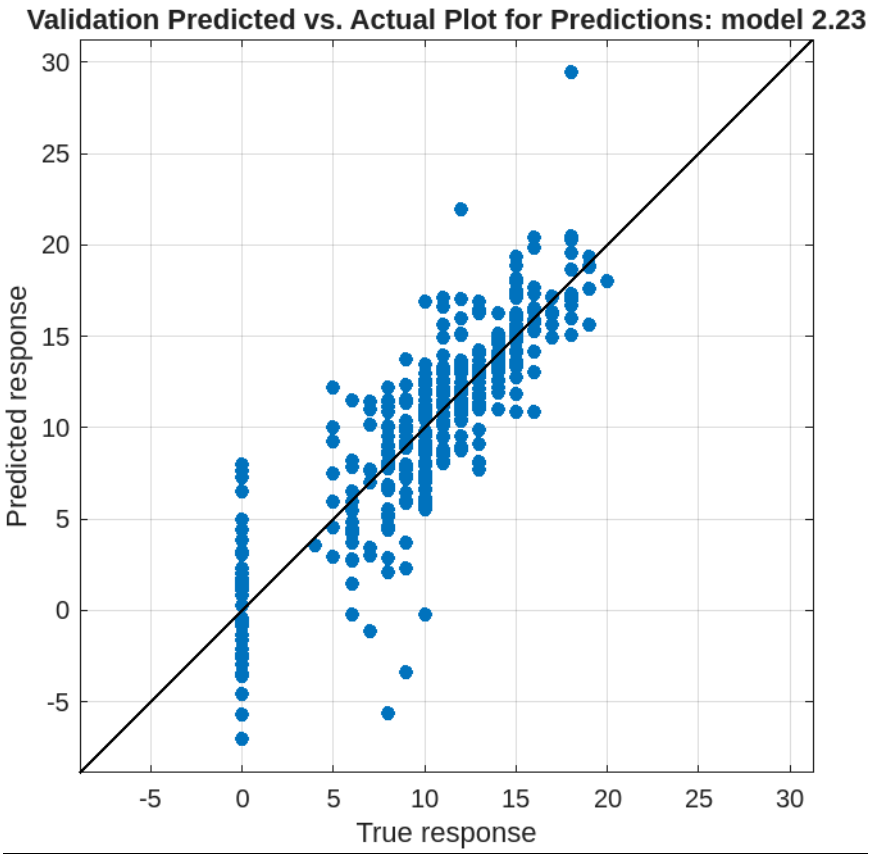
Rational quadratic GPR



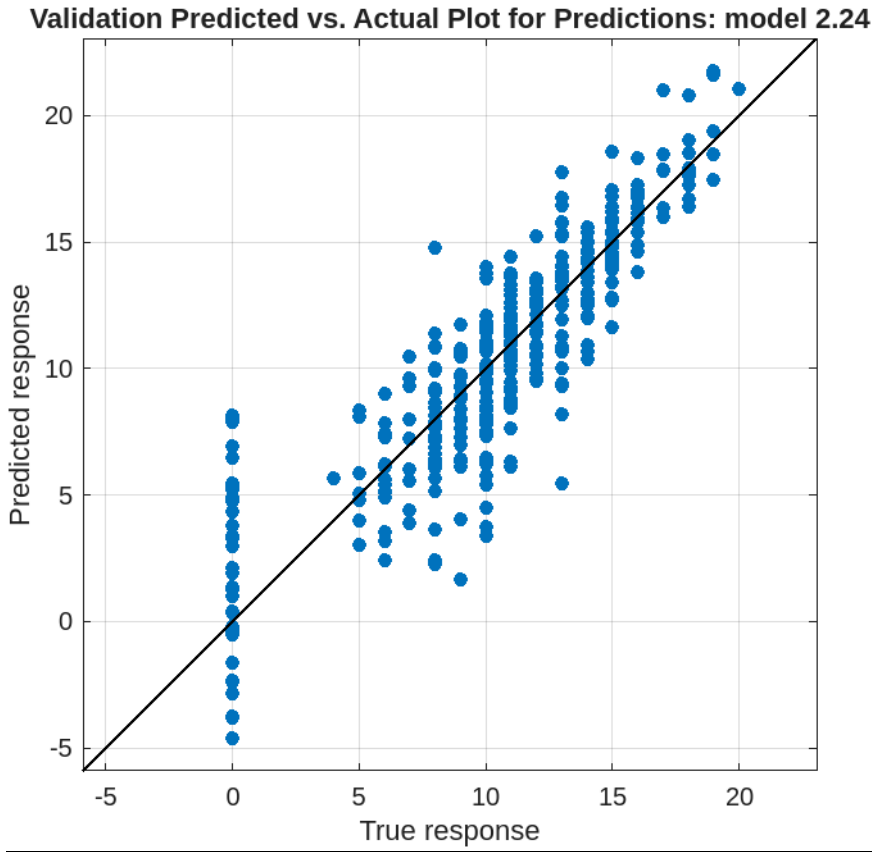
Narrow Neural Network



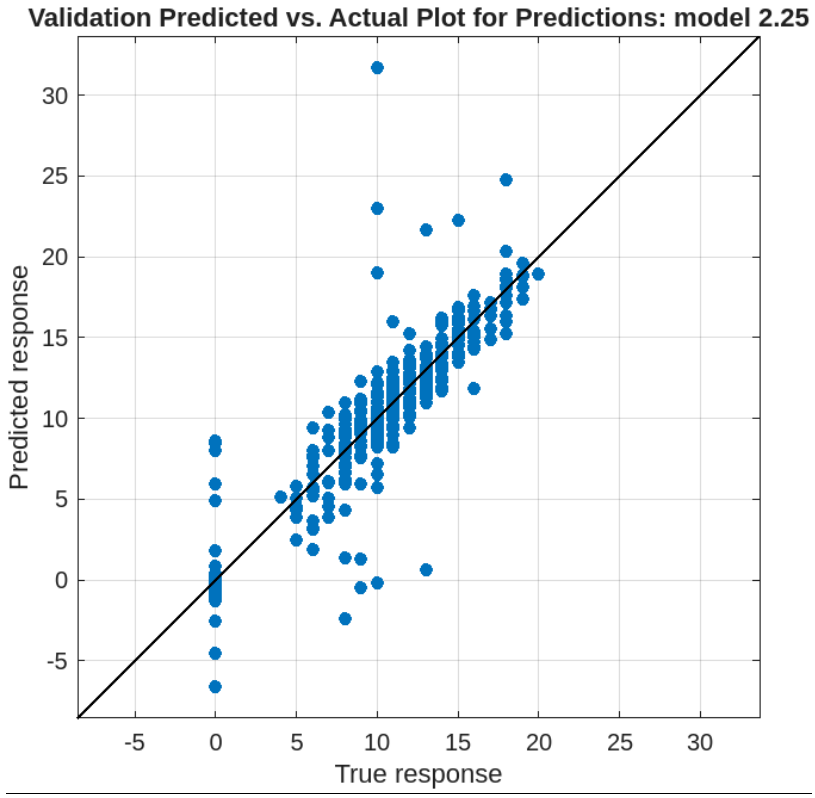
Medium Neural Network



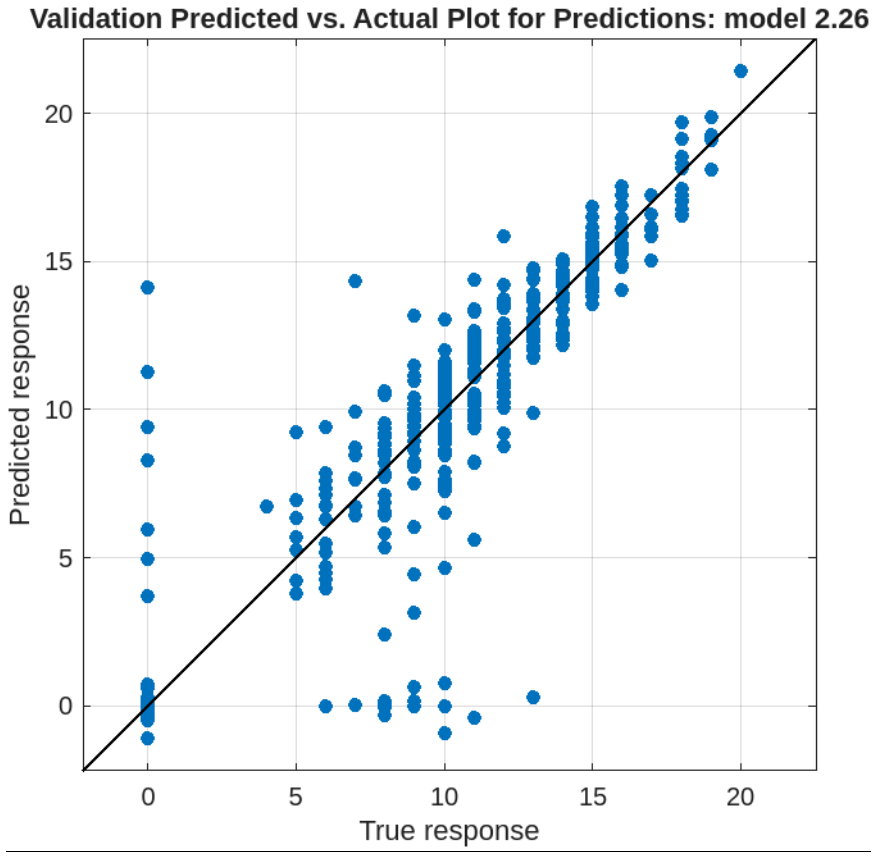
Wide Neural Network



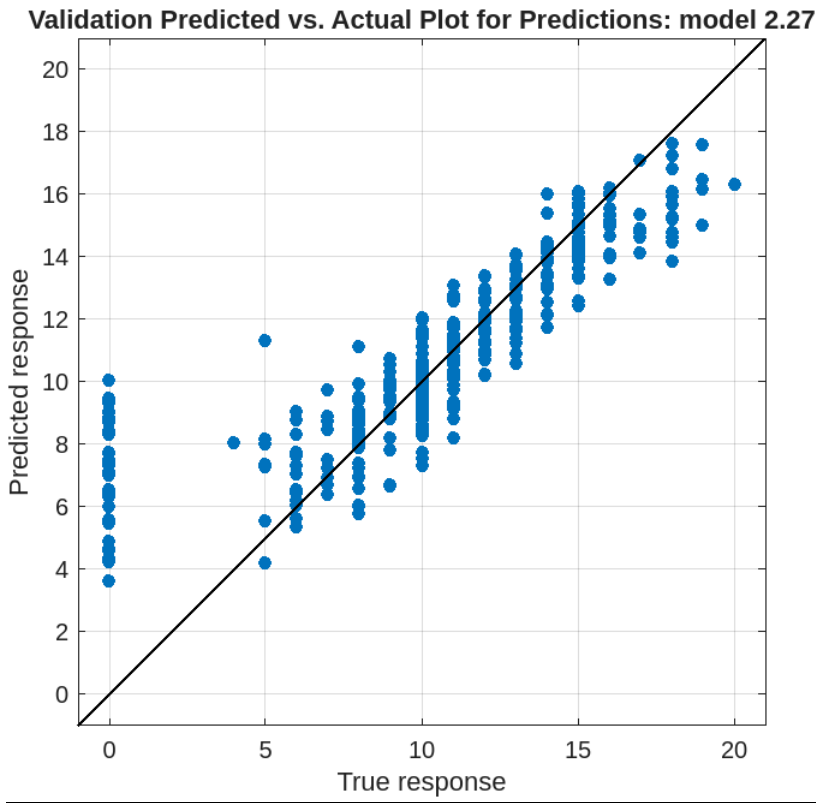
Bilayered neural network



Trilayered neural network



SVM Kernel



Least Squares Regression Kernel