Predicting Students' Academic Success Using Socioeconomic Factors

Roopesh Mikkilineni
Computer Science
Georgia State University
Atlanta, United States
rmikkilineni1@student.gsu.edu

Jessica Haines
Computer Science
Georgia State University
Atlanta, United States
jhaines4@student.gsu.edu

Ryan Ferguson
Computer Science
Georgia State University
Atlanta, United States
rferguson11@student.gsu.edu

Abhinav Kompella
Computer Science
Georgia State University
Atlanta, United States
akompella1@student.gsu.edu

***Abstract*—This is the Machine Learning project for Dr. Banda's Machine Learning Class Spring Semester 2023.**

Our project link:
https://github.com/kompellabhinav/CSC4850-Machine-Learning-FInal-Project

I. Introduction

Higher education institutions collect a lot of data regarding their students that can be used to generate even more information for monitoring their success. Academic success is important for those that wish to grow either economically or in their careers. As such, dropout is one of the most problematic issues that all higher education institutes need to address to improve their success. There are many different factors that can lead a student to dropout such as demographics, socioeconomic factors and macroeconomic factors. This report highlights the most prominent factors that result in a student dropping out. In this report we further discuss: (1) our project idea, (2) an extended survey of related work, (3) our data source, (4) key algorithms/methods that we used, (5) how the algorithms were implemented, (6) interesting observations, (7) and the key results from our work.

II. Project Idea

For our project we built machine learning models that use social and economic factors of students to predict whether a student drops out or not. The reasons we chose this topic are:

1. A large percentage of students dropped this class, and we thought this topic was both interesting and relevant.
2. As students we know that a majority of our willingness to stay in a class comes from the difficulty of coursework and the instructor's teaching style, but we were curious about external factors.
3. Dropping out is often judged to be a consequence of incompetence or lack of determination. We wanted to challenge this narrative by investigating a specific category of variables that are often overlooked.
4. We found that even in the most educationally successful countries like Denmark, there is still only an eighty percent graduation rate.

One thing to note is that, while a dropout is usually equated with a lack of degree, for the scope of this project dropping out includes transferring between fields and institutions. This allows us to examine more relevant activity than just if a student completely left an institution.

III. Survey of Related Work

This topic is a very widely researched topic. Predicting how a student would perform is one of the important criteria for college admissions. Even though many studies have been performed on this data, each study differs based on how dropout is defined. Whether dropout is defined strictly as those who stop attending or along with those who transfer, the study can change greatly. Due to this change in results, very few studies can be compared directly with each other. Out of multiple research studies, the six Machine Learning models mainly used were: Decision Tree, artificial neural networks, SVMs, K-Nearest Neighbor, Linear Regression, and Naive Bayes [1]. We cover most of these models in our paper as well. We also choose which of these models performs best with predicting how the data is classified.

The kind of environment the student grows in has a huge effect on his/her chance of giving higher performance in college. ML is a method for finding hidden information by studying numerous data sources in fields including business, social, medical, and education. The more data there is, such as in large databases, the better the forecast [2]. Through this project we aimed to use the Machine Learning algorithms to see how such hidden features affect the student performance.

IV. Our Data Source

The data source we chose was found on Kaggle, but it was originally gathered by four researchers from the Polytechnic Institute of Portalegre[3]. It is a dataset of demographic data of students and many attributes relating to their status in their higher education program. The data is from enrolled students between 2008 and 2019. It has 35 attributes, 4424 entries, no null values, and all non-target features are numerical:

```
Data columns (total 35 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Marital status                                  4424 non-null   int64
 1   Application mode                                4424 non-null   int64
 2   Application order                               4424 non-null   int64
 3   Course                                          4424 non-null   int64
 4   Daytime/evening attendance                      4424 non-null   int64
 5   Previous qualification                          4424 non-null   int64
 6   Nacionality                                     4424 non-null   int64
 7   Mother's qualification                          4424 non-null   int64
 8   Father's qualification                          4424 non-null   int64
 9   Mother's occupation                             4424 non-null   int64
 10  Father's occupation                             4424 non-null   int64
 11  Displaced                                       4424 non-null   int64
 12  Educational special needs                       4424 non-null   int64
 13  Debtor                                          4424 non-null   int64
 14  Tuition fees up to date                         4424 non-null   int64
 15  Gender                                          4424 non-null   int64
 16  Scholarship holder                              4424 non-null   int64
 17  Age at enrollment                               4424 non-null   int64
 18  International                                   4424 non-null   int64
 19  Curricular units 1st sem (credited)             4424 non-null   int64
 20  Curricular units 1st sem (enrolled)             4424 non-null   int64
 21  Curricular units 1st sem (evaluations)          4424 non-null   int64
 22  Curricular units 1st sem (approved)             4424 non-null   int64
 23  Curricular units 1st sem (grade)                4424 non-null   float64
 24  Curricular units 1st sem (without evaluations)  4424 non-null   int64
 25  Curricular units 2nd sem (credited)             4424 non-null   int64
 26  Curricular units 2nd sem (enrolled)             4424 non-null   int64
 27  Curricular units 2nd sem (evaluations)          4424 non-null   int64
 28  Curricular units 2nd sem (approved)             4424 non-null   int64
 29  Curricular units 2nd sem (grade)                4424 non-null   float64
 30  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 31  Unemployment rate                               4424 non-null   float64
 32  Inflation rate                                  4424 non-null   float64
 33  GDP                                             4424 non-null   float64
 34  Target                                          4424 non-null   object
```

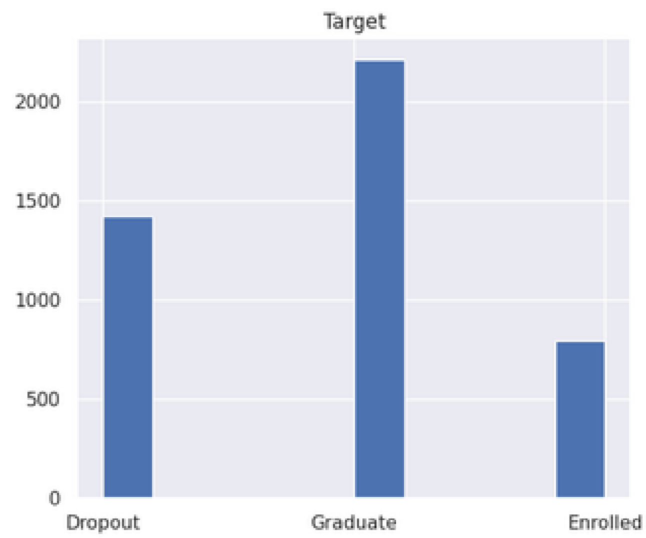
Attributes include demographics, social-economic factors, academic performance, and enrollment status at the end of each semester. It was interesting to see which of these factors were best for predicting student dropout or success. Also since there are no missing values we did not have to worry about potential errors from removing or completing incomplete entries. Even though the dataset is complete, it should be known that in the interest of further research into this topic, there needs to be a broader collection of data in both quality and volume. It is important to note that this dataset has three classes: graduate, dropout, and enrolled.

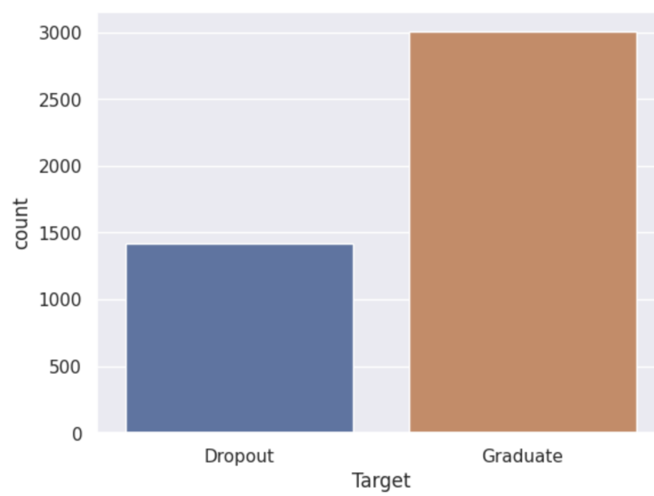
V. Preprocessing

Printing out the dataset, a description of the dataset, and a histogram for each column let us discover that a lot of the features have a skewed distribution and we were concerned this would negatively affect our models. After some testing, we discovered that the models were performing relatively well. Because of this we had very few pre-processing steps.

We did, however, round up the 'Curricular units 1st sem (grade)' and 'Curricular units 2nd sem (grade)' columns because some of the entries had 13 decimal places while most had no more than 3.

The most important processing step we took is reducing the number of targets from 3 to 2 since we are only assesing if students dropped out or not. We did this by combining the Graduated and Enrolled targets into one called Graduate:





.

VI. Key Algorithms / Methodology

The following are the algorithms we used in this project and how they were implemented to show the results.

- *A. Decision Tree*
- *B.* Random Forest
- *C.* Perceptron
- *D. Naive Bayes*
- *E. Logistic Regression*
- *F.* Linear Regression
- *G. SVM-Linear Kernel*
- *H. SVM-RBF Kernel*
- *I. Gradient Boosting*
- *J. Multi-Layer Perceptron*
- *K. K Nearest Neighbor*
- *L. SVM-Poly Kernel*
- *M. Ada Boost*

We employed k-fold cross validation, with k=10, in combination with learning curves and performance metrics to compare our models and help us choose the best one. To do this we set up a robust code framework that let us change a few parameters such as the number of folds (k), training/testing split, and type of model.

Intra-model comparisons were done first to determine the best fold and split for each model and then inter-model comparisons were done to determine the best performing model overall. These comparisons were done with advanced metrics like recall, precision and f-measure combined with learning curves and confusion matrices.

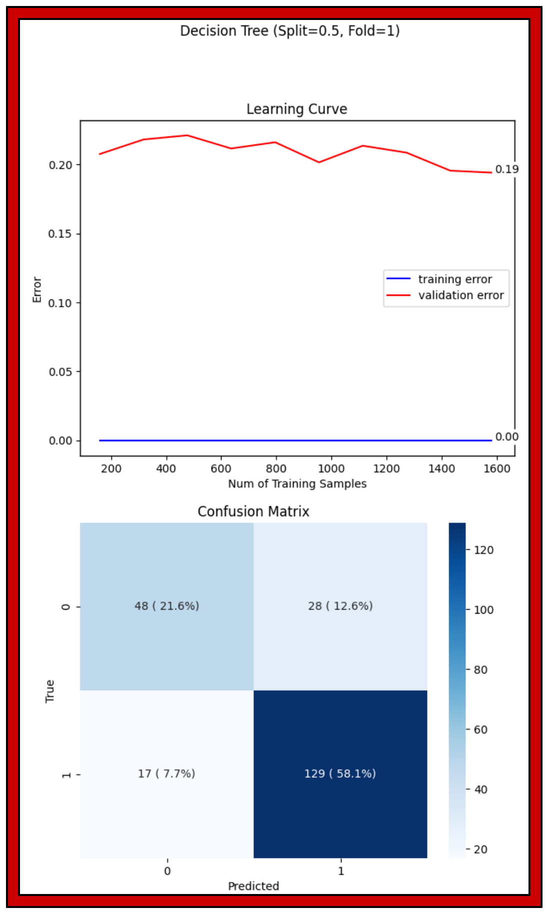
For each model we did 10 folds. For each fold we had 3 splits: 50/50, 70/30, and 80/20. We hand-picked the best fold for each split using the f1-scores. f1-score was the anchoring metrics to compare models since the metric is particularly useful for imbalance datasets such as ours. The best split was chosen based on the learning curves and generalization error.

The following plots describe each model's performance on each fold in each split. The best fold is denoted by a red box and below it is the learning curves and confusion matrix for that fold. The best split is then denoted in the same way. One thing to note is no parameter optimization was done and all the models retained their default settings.

A. Decision Tree

50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.810	0.797	0.802	0.797
1	0.841	0.842	0.840	0.842
2	0.778	0.783	0.779	0.783
3	0.803	0.801	0.802	0.801
4	0.837	0.837	0.837	0.837
5	0.789	0.792	0.790	0.792
6	0.804	0.796	0.799	0.796
7	0.757	0.756	0.756	0.756
8	0.805	0.801	0.802	0.801
9	0.822	0.814	0.818	0.814



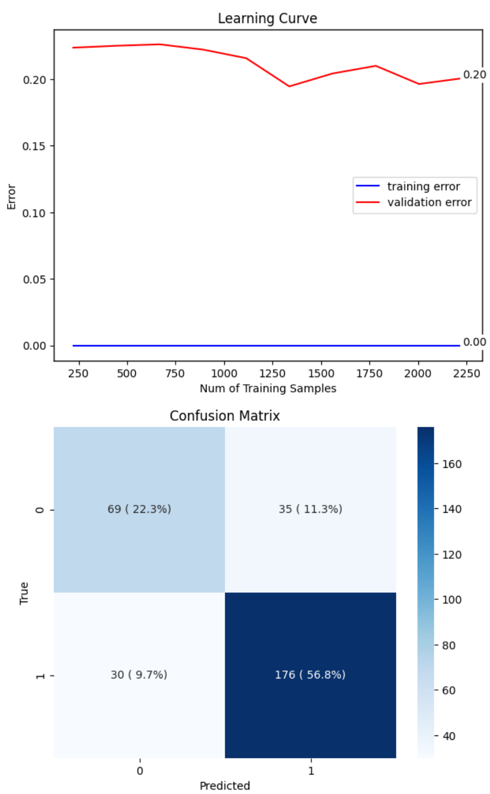
70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.776	0.777	0.776	0.777
1	0.793	0.790	0.792	0.790
2	0.830	0.819	0.823	0.819
3	0.809	0.813	0.810	0.813
4	0.809	0.806	0.808	0.806
5	0.808	0.810	0.809	0.810
6	0.811	0.806	0.808	0.806
7	0.818	0.806	0.809	0.806
8	0.798	0.799	0.799	0.799
9	0.740	0.741	0.741	0.741

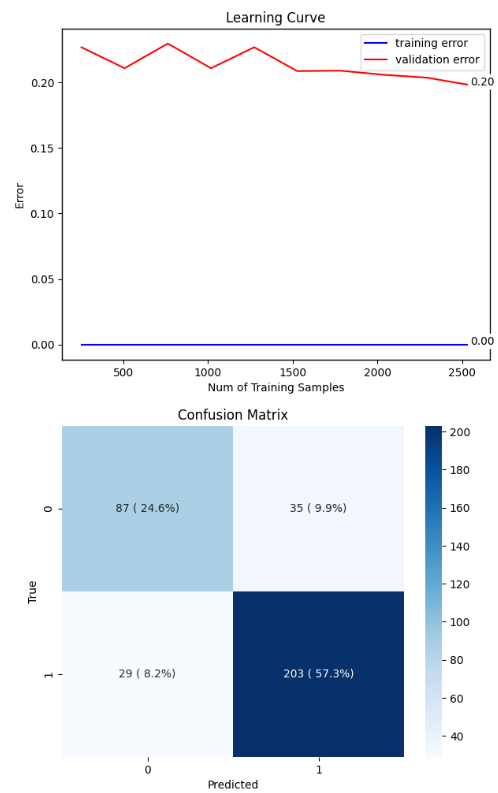
80/20 Split

	Precision	Recall	F1score	Accuracy
0	0.794	0.794	0.794	0.794
1	0.808	0.811	0.809	0.811
2	0.804	0.802	0.803	0.802
3	0.807	0.805	0.806	0.805
4	0.789	0.788	0.789	0.788
5	0.805	0.802	0.803	0.802
6	0.822	0.819	0.820	0.819
7	0.824	0.819	0.821	0.819
8	0.754	0.754	0.754	0.754
9	0.766	0.762	0.764	0.762

Decision Tree (Split=0.3, Fold=2)



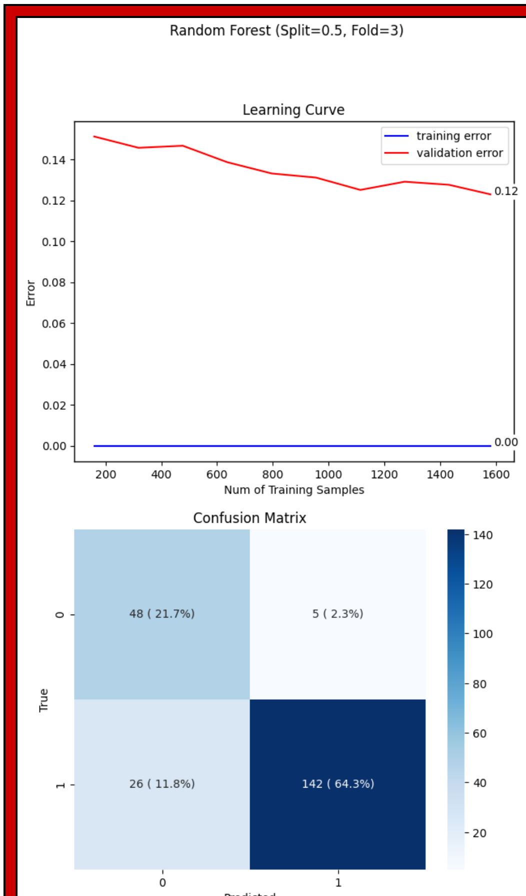
Decision Tree (Split=0.2, Fold=7)



B. *Random Forest*

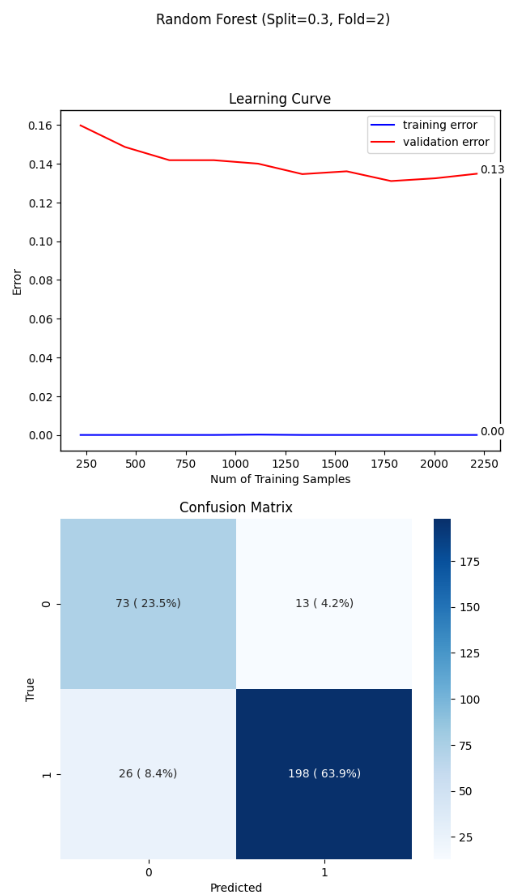
50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.887	0.887	0.883	0.887
1	0.883	0.874	0.868	0.874
2	0.865	0.860	0.853	0.860
3	0.909	0.910	0.908	0.910
4	0.862	0.860	0.856	0.860
5	0.842	0.842	0.837	0.842
6	0.857	0.860	0.857	0.860
7	0.869	0.869	0.866	0.869
8	0.886	0.887	0.884	0.887
9	0.854	0.855	0.854	0.855



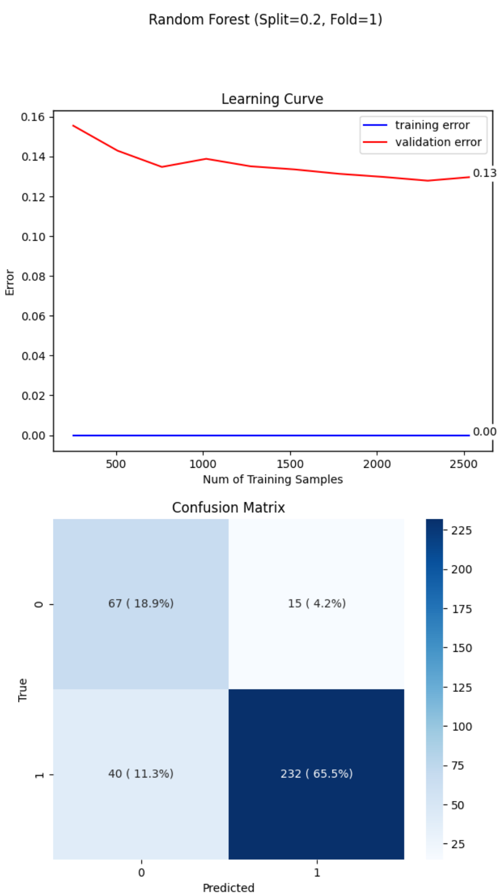
70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.855	0.855	0.849	0.855
1	0.873	0.874	0.872	0.874
2	0.902	0.903	0.903	0.903
3	0.873	0.871	0.865	0.871
4	0.862	0.865	0.863	0.865
5	0.872	0.868	0.863	0.868
6	0.852	0.854	0.851	0.854
7	0.883	0.883	0.883	0.883
8	0.842	0.841	0.835	0.841
9	0.842	0.841	0.836	0.841



80/20 Split

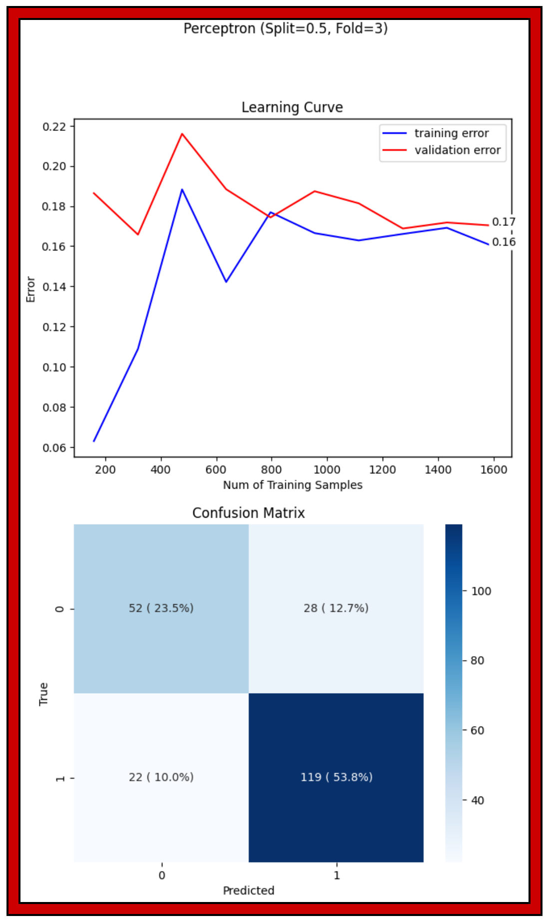
	Precision	Recall	F1score	Accuracy
0	0.842	0.845	0.838	0.845
1	0.899	0.895	0.892	0.895
2	0.872	0.873	0.869	0.873
3	0.862	0.864	0.861	0.864
4	0.862	0.862	0.859	0.862
5	0.890	0.887	0.884	0.887
6	0.889	0.890	0.888	0.890
7	0.886	0.887	0.887	0.887
8	0.856	0.853	0.848	0.853
9	0.839	0.841	0.837	0.841



C. Perceptron

50/50 Split

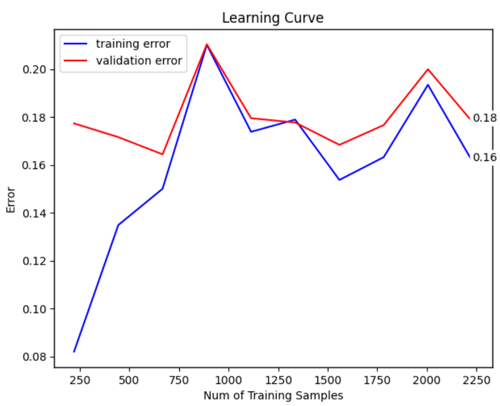
	Precision	Recall	F1score	Accuracy
0	0.861	0.860	0.861	0.860
1	0.845	0.842	0.836	0.842
2	0.779	0.774	0.776	0.774
3	0.871	0.873	0.872	0.873
4	0.869	0.869	0.869	0.869
5	0.775	0.778	0.770	0.778
6	0.842	0.837	0.839	0.837
7	0.802	0.805	0.801	0.805
8	0.799	0.783	0.788	0.783
9	0.805	0.810	0.807	0.810

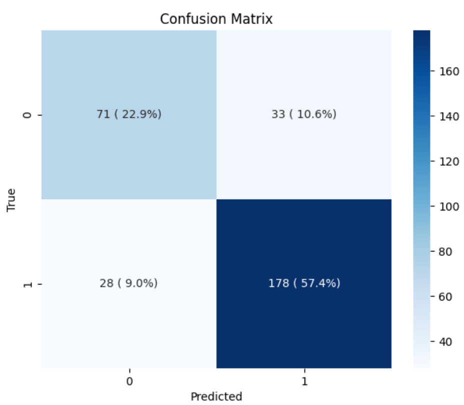


70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.806	0.803	0.804	0.803
1	0.876	0.874	0.875	0.874
2	0.858	0.855	0.856	0.855
3	0.798	0.800	0.799	0.800
4	0.766	0.745	0.752	0.745
5	0.804	0.806	0.804	0.806
6	0.787	0.790	0.788	0.790
7	0.830	0.816	0.819	0.816
8	0.819	0.822	0.816	0.822
9	0.851	0.848	0.842	0.848

Perceptron (Split=0.3, Fold=1)

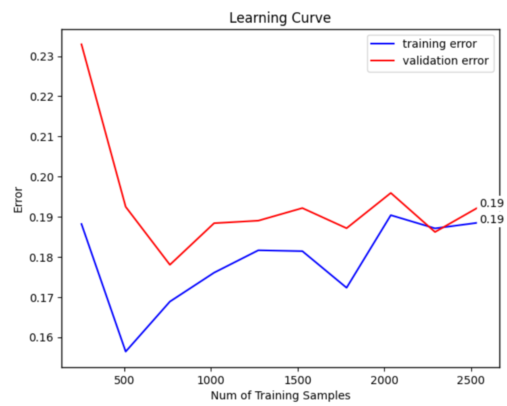


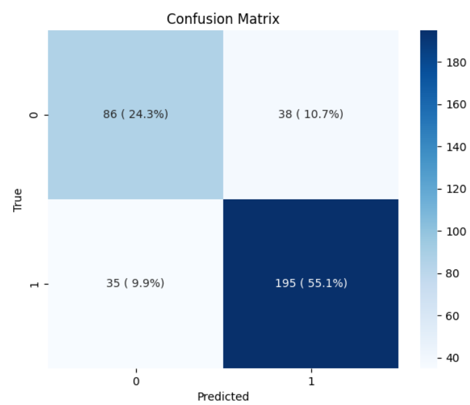


80/20 Split

	Precision	Recall	F1score	Accuracy
0	0.805	0.788	0.793	0.788
1	0.808	0.799	0.802	0.799
2	0.855	0.842	0.845	0.842
3	0.786	0.774	0.778	0.774
4	0.805	0.808	0.805	0.808
5	0.795	0.794	0.794	0.794
6	0.872	0.873	0.872	0.873
7	0.829	0.831	0.829	0.831
8	0.821	0.822	0.822	0.822
9	0.824	0.827	0.821	0.827

Perceptron (Split=0.2, Fold=6)





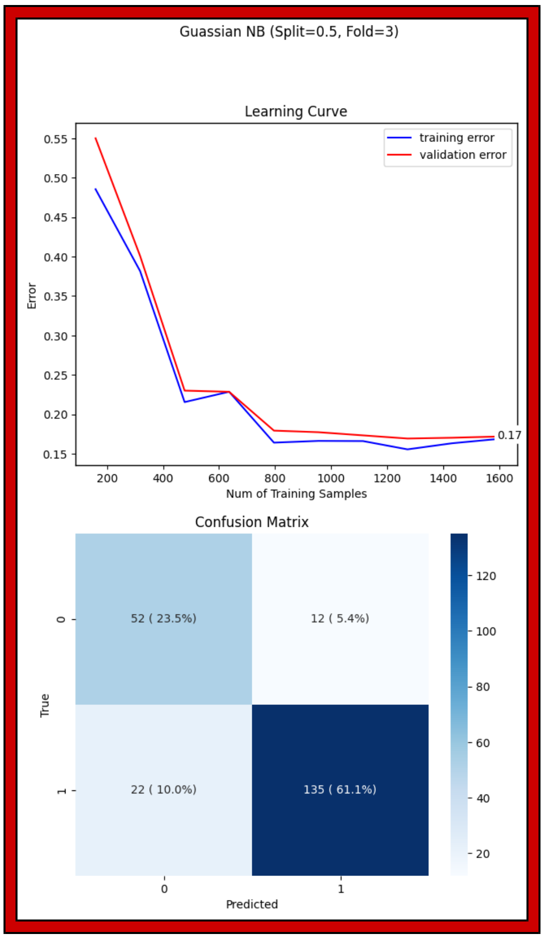
D. Naive Bayes

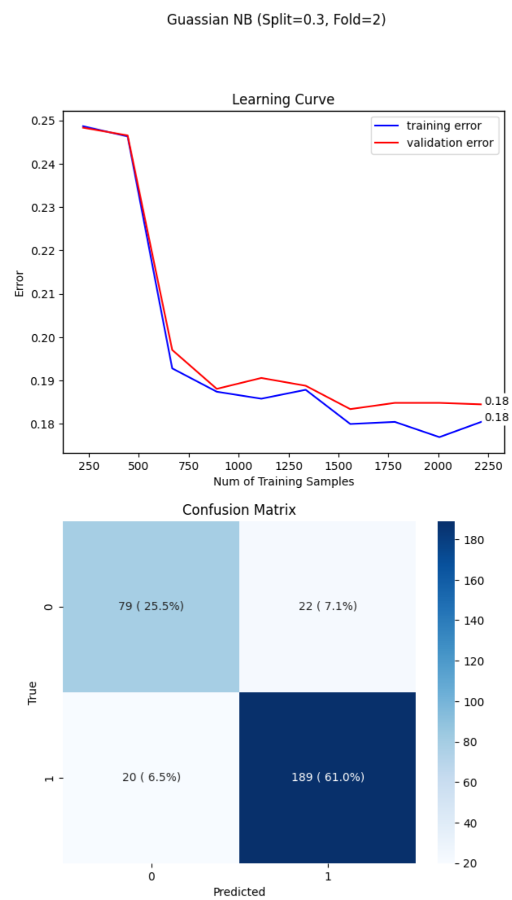
50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.836	0.833	0.834	0.833
1	0.817	0.820	0.817	0.820
2	0.844	0.846	0.843	0.846
3	0.869	0.855	0.858	0.855
4	0.845	0.846	0.845	0.846
5	0.798	0.792	0.794	0.792
6	0.835	0.833	0.834	0.833
7	0.821	0.824	0.821	0.824
8	0.851	0.851	0.851	0.851
9	0.810	0.810	0.810	0.810

70/30 Split

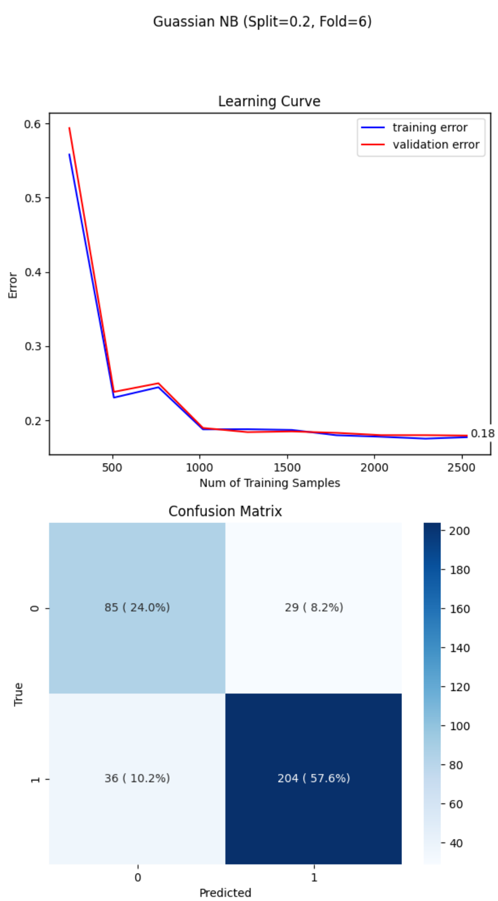
	Precision	Recall	F1score	Accuracy
0	0.794	0.797	0.795	0.797
1	0.865	0.865	0.865	0.865
2	0.870	0.871	0.871	0.871
3	0.795	0.800	0.797	0.800
4	0.854	0.852	0.852	0.852
5	0.820	0.823	0.820	0.823
6	0.781	0.777	0.779	0.777
7	0.841	0.835	0.837	0.835
8	0.837	0.838	0.838	0.838
9	0.779	0.780	0.779	0.780





80/20 Split

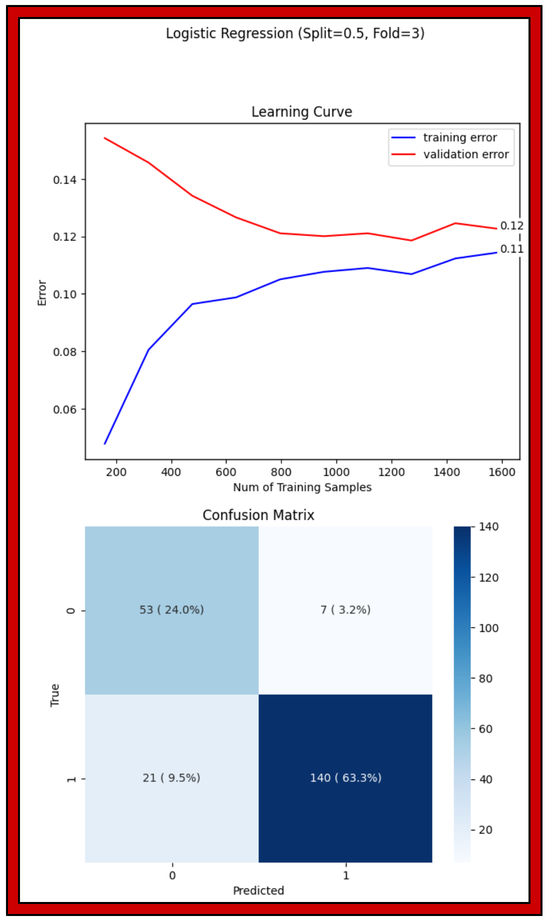
	Precision	Recall	F1score	Accuracy
0	0.836	0.839	0.837	0.839
1	0.833	0.833	0.833	0.833
2	0.782	0.777	0.779	0.777
3	0.815	0.816	0.816	0.816
4	0.846	0.847	0.846	0.847
5	0.814	0.816	0.815	0.816
6	0.850	0.850	0.850	0.850
7	0.821	0.819	0.820	0.819
8	0.823	0.825	0.823	0.825
9	0.800	0.799	0.800	0.799



E. *Logistic Regression*

50/50 Split

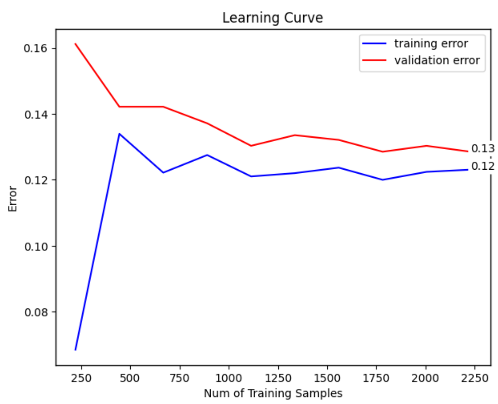
	Precision	Recall	F1score	Accuracy
0	0.891	0.892	0.889	0.892
1	0.885	0.878	0.874	0.878
2	0.874	0.873	0.870	0.873
3	0.913	0.914	0.913	0.914
4	0.881	0.878	0.875	0.878
5	0.850	0.851	0.848	0.851
6	0.871	0.873	0.870	0.873
7	0.871	0.869	0.865	0.869
8	0.896	0.896	0.893	0.896
9	0.859	0.860	0.859	0.860

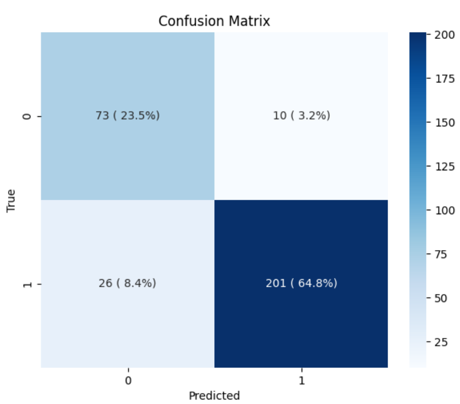


70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.870	0.868	0.862	0.868
1	0.884	0.884	0.881	0.884
2	0.912	0.913	0.912	0.913
3	0.858	0.855	0.846	0.855
4	0.852	0.855	0.852	0.855
5	0.868	0.868	0.865	0.868
6	0.859	0.861	0.859	0.861
7	0.880	0.880	0.880	0.880
8	0.874	0.874	0.871	0.874
9	0.852	0.851	0.847	0.851

Logistic Regression (Split=0.3, Fold=2)

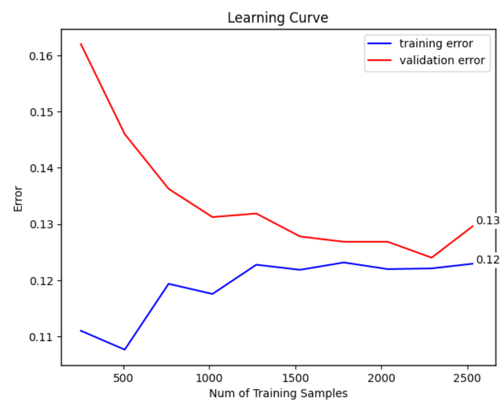


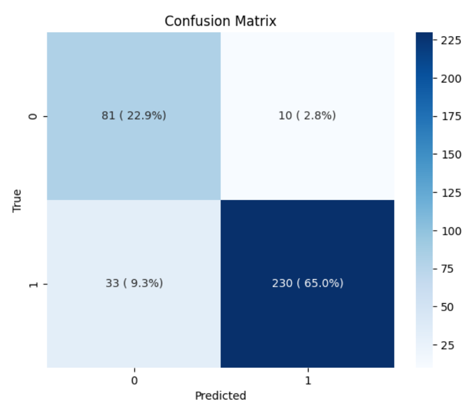


80/20 Split

	Precision	Recall	F1score	Accuracy
0	0.864	0.864	0.859	0.864
1	0.880	0.879	0.874	0.879
2	0.888	0.890	0.888	0.890
3	0.857	0.859	0.853	0.859
4	0.865	0.864	0.862	0.864
5	0.895	0.893	0.890	0.893
6	0.881	0.881	0.879	0.881
7	0.869	0.870	0.869	0.870
8	0.857	0.856	0.852	0.856
9	0.832	0.836	0.832	0.836

Logistic Regression (Split=0.2, Fold=2)

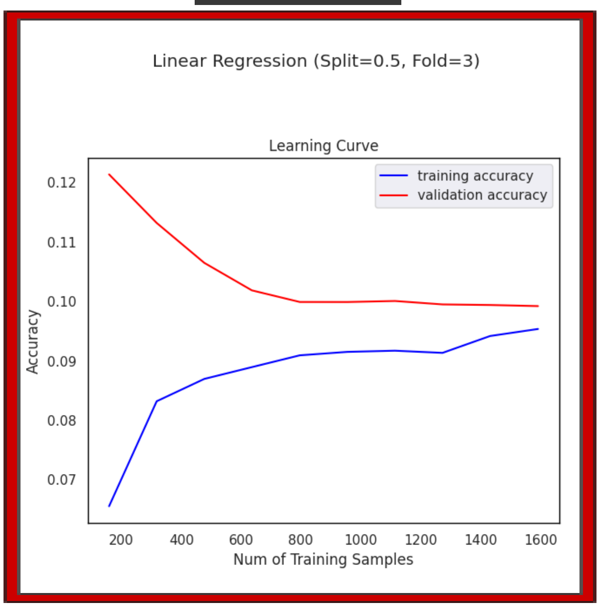




F. Linear Regression

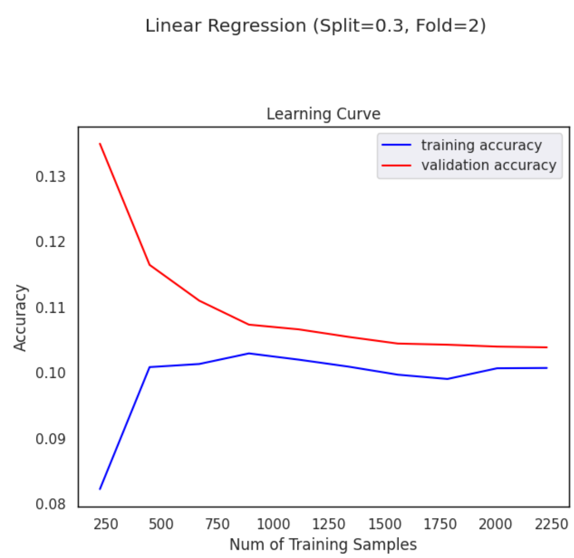
50/50 Split

	RMSE	r2
0	0.291	0.591
1	0.324	0.538
2	0.324	0.528
3	0.276	0.643
4	0.305	0.601
5	0.338	0.504
6	0.313	0.527
7	0.321	0.548
8	0.313	0.540
9	0.341	0.411



70/30 Split

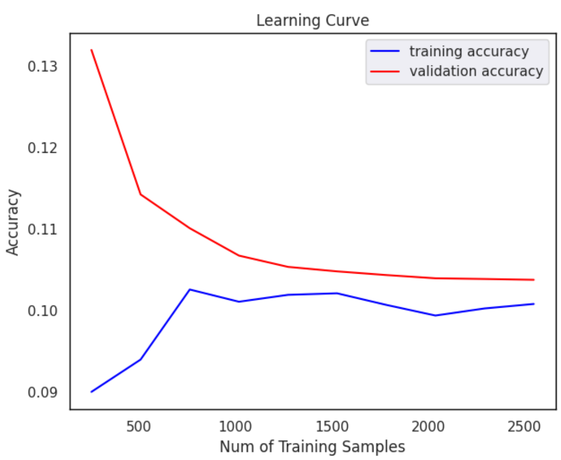
	RMSE	r2
0	0.332	0.493
1	0.304	0.574
2	0.282	0.633
3	0.329	0.490
4	0.325	0.503
5	0.328	0.531
6	0.332	0.495
7	0.307	0.579
8	0.323	0.529
9	0.339	0.497



80/20 Split

	RMSE	r2
0	0.329	0.488
1	0.311	0.558
2	0.307	0.561
3	0.325	0.493
4	0.314	0.573
5	0.313	0.563
6	0.305	0.578
7	0.316	0.557
8	0.352	0.459
9	0.338	0.475

Linear Regression (Split=0.2, Fold=6)

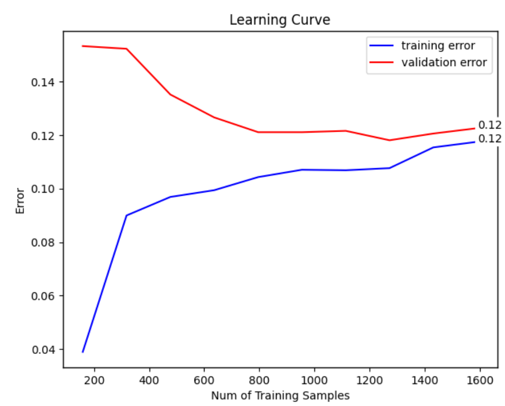


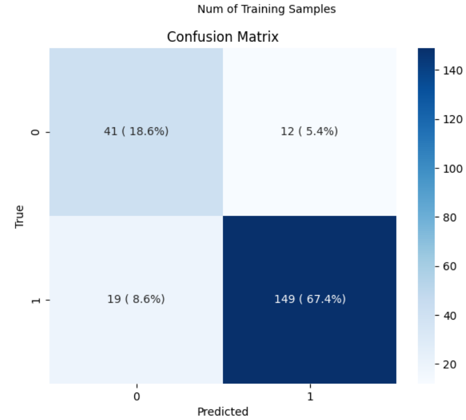
G. *SVM-Linear Kernel*

50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.901	0.901	0.898	0.901
1	0.871	0.865	0.859	0.865
2	0.862	0.855	0.848	0.855
3	0.891	0.891	0.889	0.891
4	0.867	0.860	0.854	0.860
5	0.865	0.860	0.855	0.860
6	0.876	0.878	0.874	0.878
7	0.878	0.873	0.869	0.873
8	0.894	0.891	0.887	0.891
9	0.856	0.860	0.857	0.860

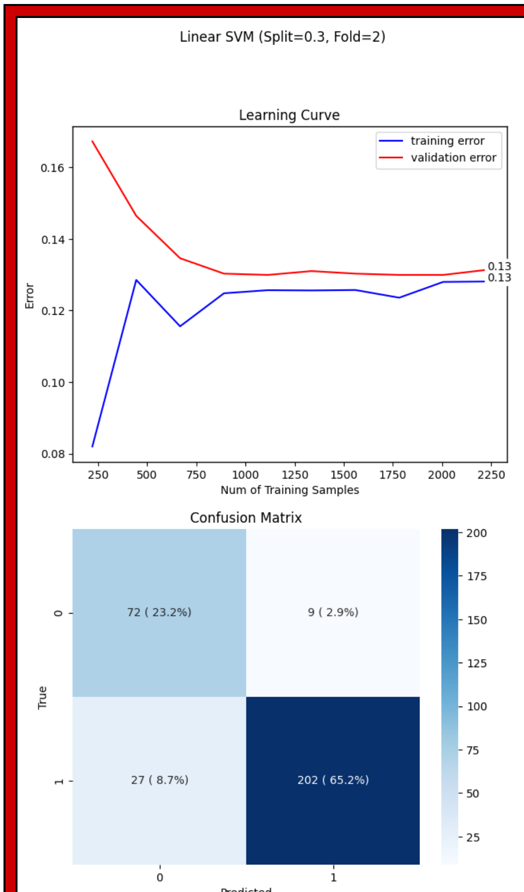
Linear SVM (Split=0.5, Fold=0)





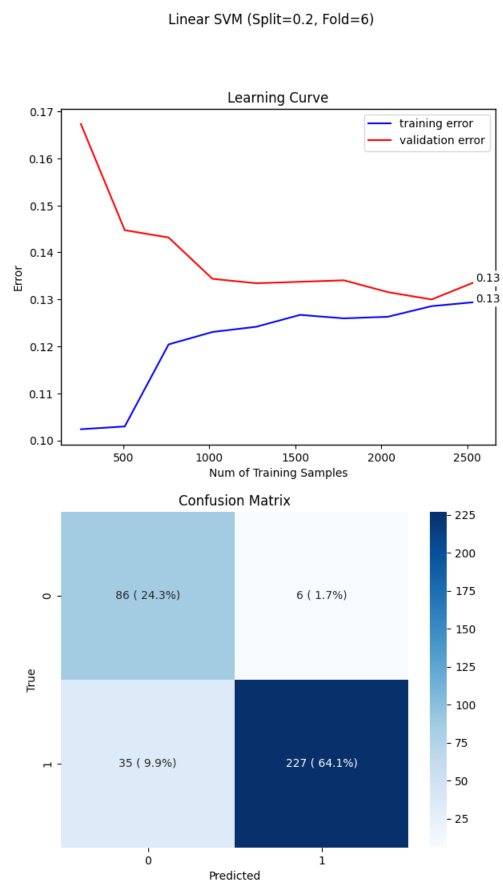
70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.842	0.839	0.829	0.839
1	0.884	0.884	0.880	0.884
2	0.923	0.923	0.921	0.923
3	0.849	0.848	0.840	0.848
4	0.852	0.855	0.852	0.855
5	0.862	0.858	0.853	0.858
6	0.882	0.880	0.876	0.880
7	0.892	0.893	0.892	0.893
8	0.880	0.880	0.877	0.880
9	0.850	0.845	0.838	0.845



80/20 Split

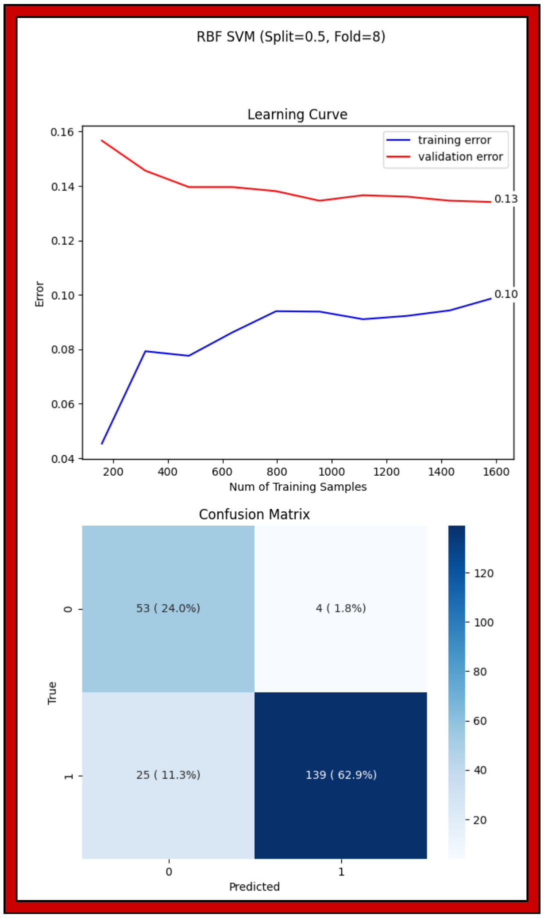
	Precision	Recall	F1score	Accuracy
0	0.861	0.859	0.851	0.859
1	0.893	0.887	0.882	0.887
2	0.868	0.870	0.867	0.870
3	0.856	0.859	0.854	0.859
4	0.865	0.862	0.857	0.862
5	0.890	0.884	0.880	0.884
6	0.887	0.887	0.884	0.887
7	0.869	0.870	0.868	0.870
8	0.851	0.850	0.846	0.850
9	0.839	0.839	0.831	0.839



H. SVM-RBF Kernel

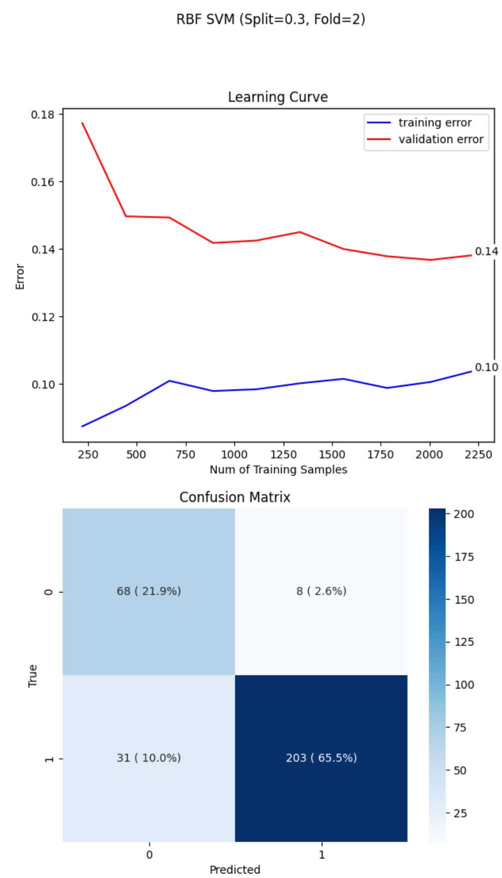
50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.894	0.892	0.887	0.892
1	0.859	0.851	0.844	0.851
2	0.879	0.869	0.861	0.869
3	0.890	0.891	0.889	0.891
4	0.880	0.864	0.857	0.864
5	0.835	0.833	0.827	0.833
6	0.878	0.878	0.873	0.878
7	0.877	0.869	0.863	0.869
8	0.898	0.896	0.892	0.896
9	0.837	0.842	0.837	0.842



70/30 Split

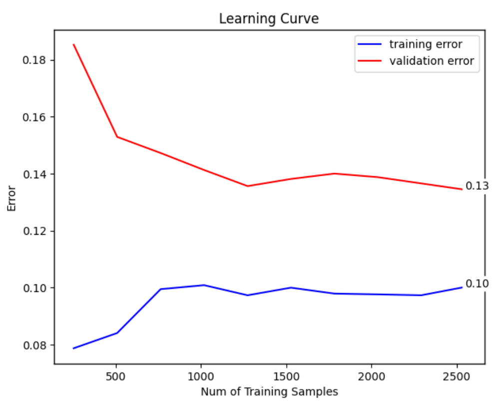
	Precision	Recall	F1score	Accuracy
0	0.839	0.839	0.830	0.839
1	0.876	0.874	0.869	0.874
2	0.914	0.913	0.911	0.913
3	0.849	0.845	0.835	0.845
4	0.863	0.865	0.861	0.865
5	0.860	0.852	0.844	0.852
6	0.872	0.871	0.866	0.871
7	0.883	0.883	0.882	0.883
8	0.866	0.864	0.859	0.864
9	0.819	0.816	0.806	0.816

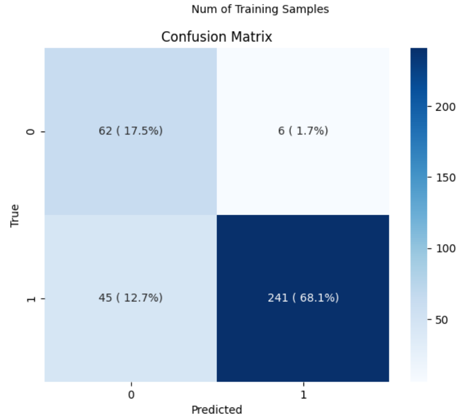


80/20 Split

	Precision	Recall	F1score	Accuracy
0	0.864	0.856	0.845	0.856
1	0.892	0.884	0.878	0.884
2	0.850	0.850	0.844	0.850
3	0.863	0.864	0.859	0.864
4	0.867	0.862	0.857	0.862
5	0.882	0.873	0.867	0.873
6	0.879	0.876	0.871	0.876
7	0.878	0.879	0.876	0.879
8	0.846	0.842	0.835	0.842
9	0.823	0.824	0.815	0.824

RBF SVM (Split=0.2, Fold=1)



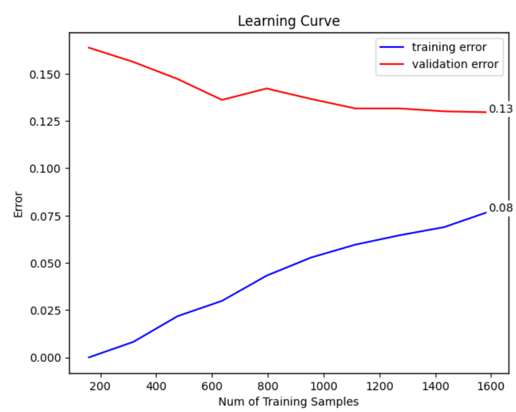


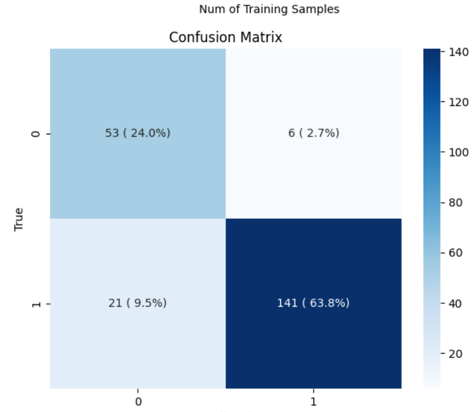
I. *Gradient Boosting*

50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.893	0.892	0.888	0.892
1	0.887	0.878	0.873	0.878
2	0.880	0.878	0.874	0.878
3	0.890	0.891	0.890	0.891
4	0.873	0.873	0.871	0.873
5	0.838	0.837	0.832	0.837
6	0.848	0.851	0.849	0.851
7	0.879	0.878	0.875	0.878
8	0.881	0.882	0.881	0.882
9	0.852	0.855	0.853	0.855

Gradient Boosting (Split=0.5, Fold=3)



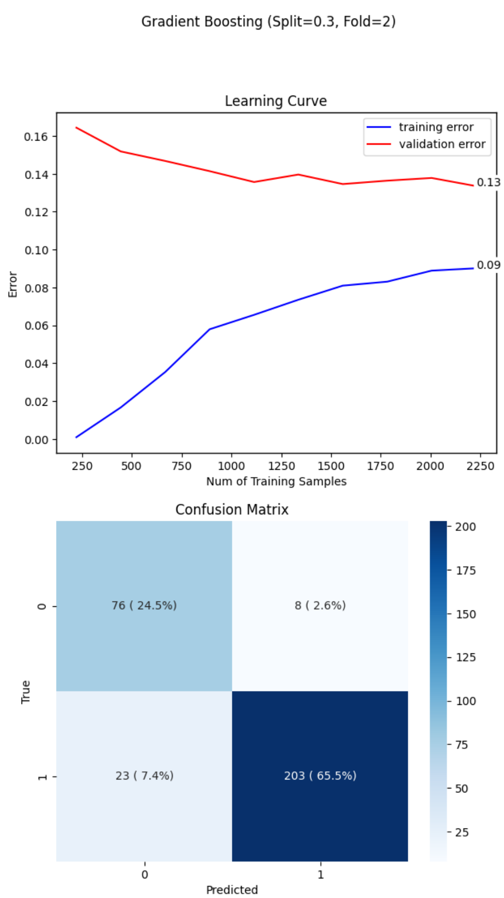


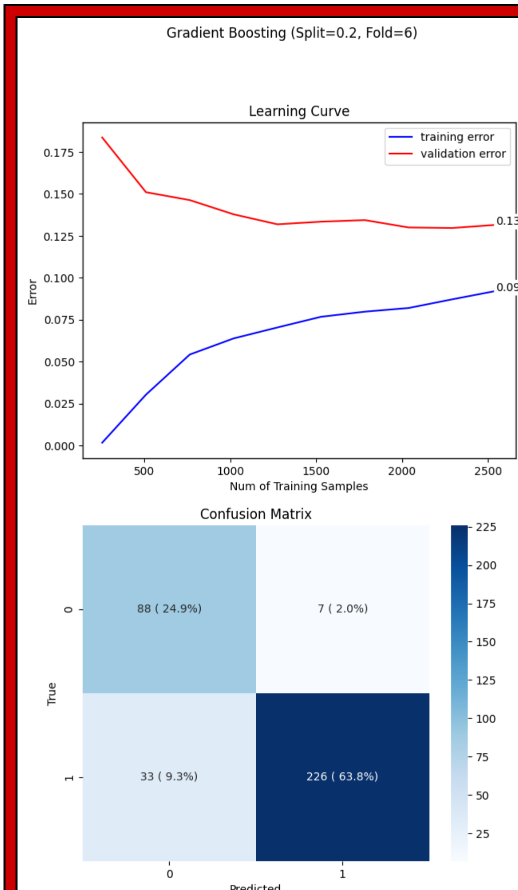
70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.871	0.871	0.867	0.871
1	0.900	0.900	0.898	0.900
2	0.922	0.923	0.922	0.923
3	0.867	0.861	0.852	0.861
4	0.859	0.861	0.860	0.861
5	0.868	0.868	0.865	0.868
6	0.863	0.864	0.861	0.864
7	0.896	0.896	0.896	0.896
8	0.867	0.867	0.864	0.867
9	0.856	0.854	0.850	0.854

80/20 Split

	Precision	Recall	F1score	Accuracy
0	0.857	0.859	0.854	0.859
1	0.893	0.890	0.886	0.890
2	0.868	0.870	0.867	0.870
3	0.868	0.870	0.866	0.870
4	0.864	0.864	0.862	0.864
5	0.891	0.887	0.883	0.887
6	0.901	0.898	0.895	0.898
7	0.895	0.895	0.895	0.895
8	0.857	0.856	0.852	0.856
9	0.836	0.839	0.833	0.839



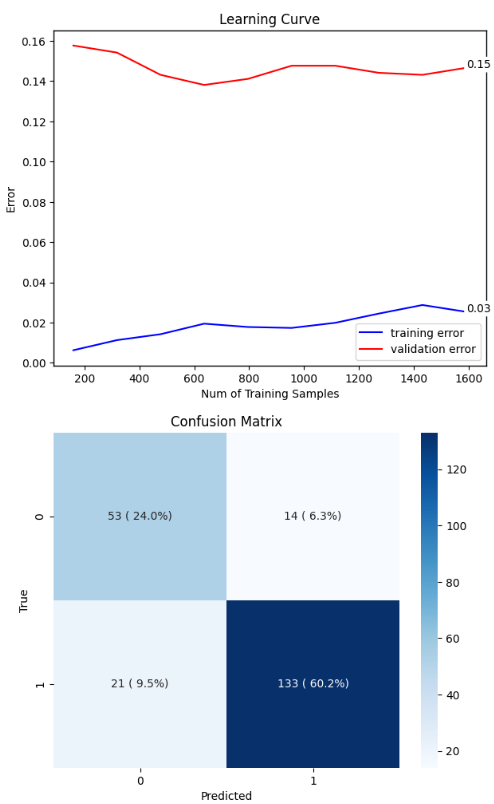


J. Multi-Layer Perceptron

50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.871	0.874	0.871	0.874
1	0.875	0.874	0.871	0.874
2	0.839	0.842	0.840	0.842
3	0.898	0.896	0.897	0.896
4	0.836	0.837	0.835	0.837
5	0.823	0.824	0.819	0.824
6	0.819	0.819	0.819	0.819
7	0.882	0.882	0.880	0.882
8	0.848	0.851	0.849	0.851
9	0.833	0.833	0.833	0.833

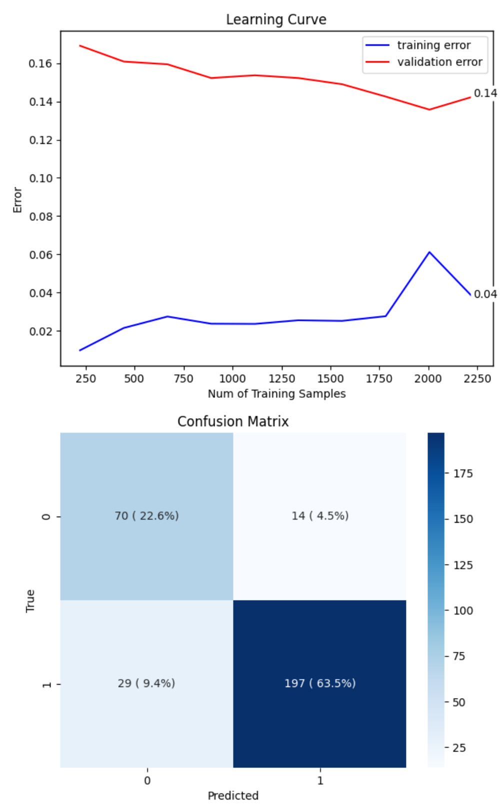
Multi Layer Perceptron (Split=0.5, Fold=3)



70/30 Split

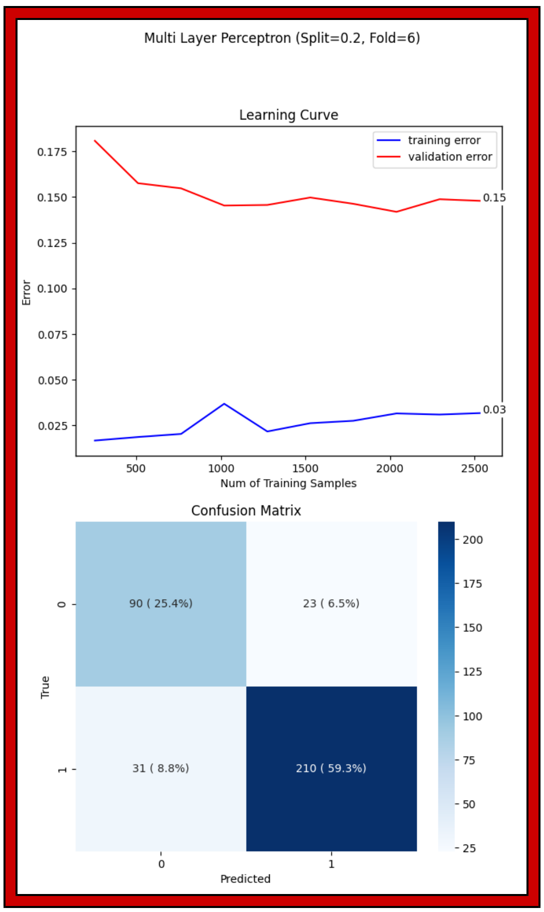
	Precision	Recall	F1score	Accuracy
0	0.867	0.868	0.864	0.868
1	0.859	0.861	0.858	0.861
2	0.919	0.919	0.919	0.919
3	0.849	0.852	0.848	0.852
4	0.850	0.852	0.851	0.852
5	0.826	0.826	0.826	0.826
6	0.876	0.877	0.876	0.877
7	0.881	0.880	0.880	0.880
8	0.860	0.861	0.857	0.861
9	0.823	0.825	0.823	0.825

Multi Layer Perceptron (Split=0.3, Fold=2)



80/20 Split

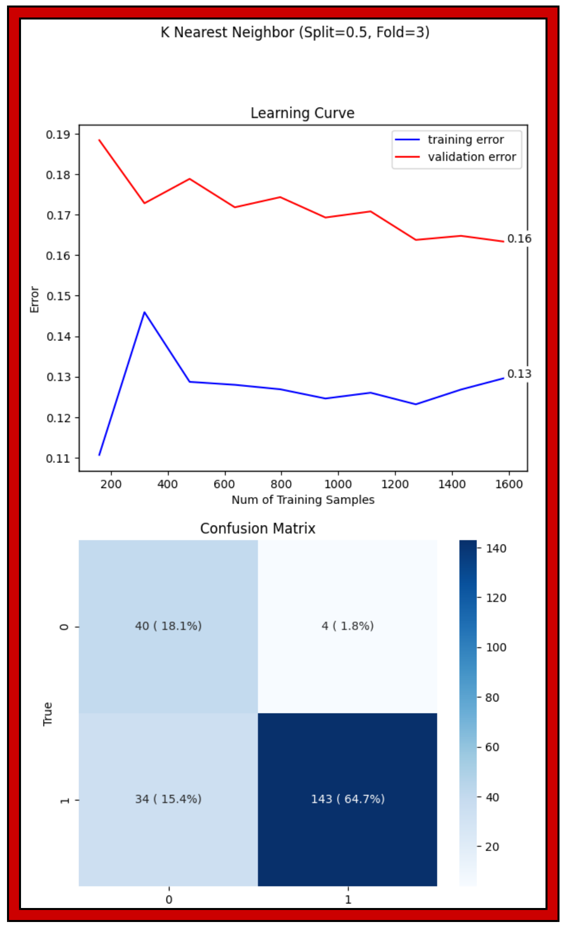
	Precision	Recall	F1score	Accuracy
0	0.848	0.850	0.845	0.850
1	0.877	0.879	0.876	0.879
2	0.869	0.867	0.868	0.867
3	0.823	0.819	0.821	0.819
4	0.840	0.842	0.840	0.842
5	0.846	0.847	0.846	0.847
6	0.889	0.890	0.888	0.890
7	0.861	0.862	0.861	0.862
8	0.832	0.833	0.830	0.833
9	0.824	0.827	0.821	0.827



K. K-Nearest Neighbor

50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.845	0.847	0.839	0.847
1	0.837	0.824	0.812	0.824
2	0.842	0.828	0.814	0.828
3	0.863	0.864	0.859	0.864
4	0.863	0.846	0.837	0.846
5	0.806	0.805	0.797	0.805
6	0.818	0.824	0.816	0.824
7	0.869	0.855	0.847	0.855
8	0.851	0.851	0.843	0.851
9	0.783	0.792	0.786	0.792



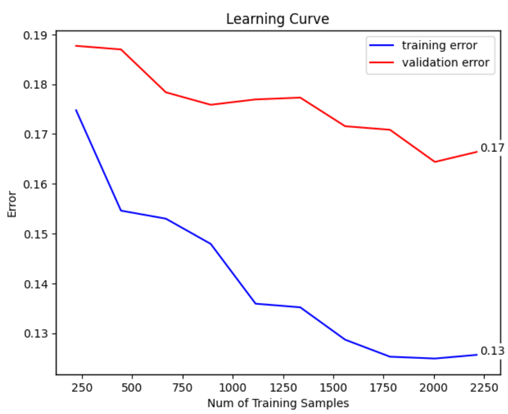
70/30 Split

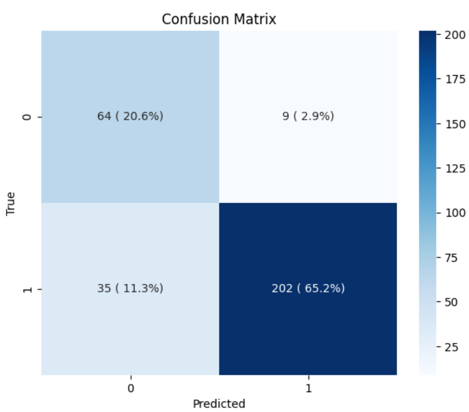
	Precision	Recall	F1score	Accuracy
0	0.841	0.835	0.824	0.835
1	0.860	0.858	0.851	0.858
2	0.882	0.881	0.876	0.881
3	0.841	0.839	0.828	0.839
4	0.833	0.835	0.828	0.835
5	0.807	0.806	0.798	0.806
6	0.823	0.825	0.818	0.825
7	0.854	0.854	0.850	0.854
8	0.841	0.838	0.830	0.838
9	0.811	0.809	0.800	0.809

80/20 Split

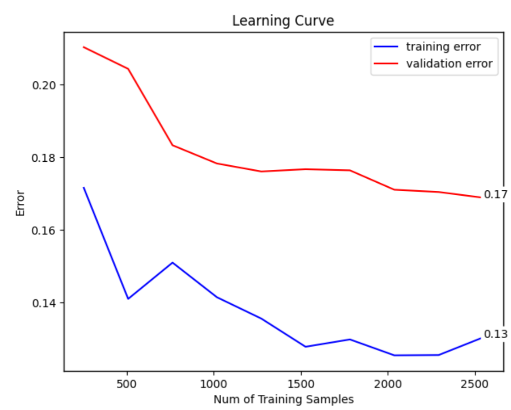
	Precision	Recall	F1score	Accuracy
0	0.831	0.831	0.819	0.831
1	0.847	0.845	0.836	0.845
2	0.831	0.833	0.825	0.833
3	0.842	0.845	0.837	0.845
4	0.835	0.833	0.828	0.833
5	0.840	0.833	0.824	0.833
6	0.836	0.836	0.829	0.836
7	0.841	0.839	0.832	0.839
8	0.832	0.828	0.820	0.828
9	0.828	0.824	0.812	0.824

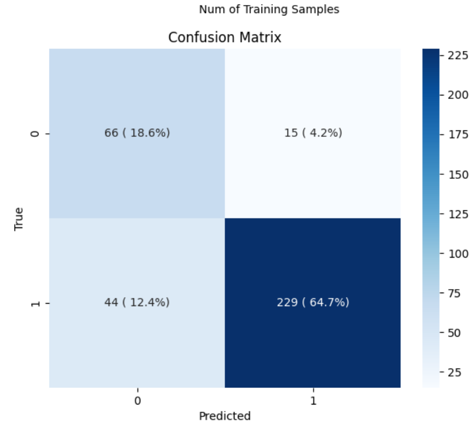
K Nearest Neighbor (Split=0.3, Fold=2)





K Nearest Neighbor (Split=0.2, Fold=3)





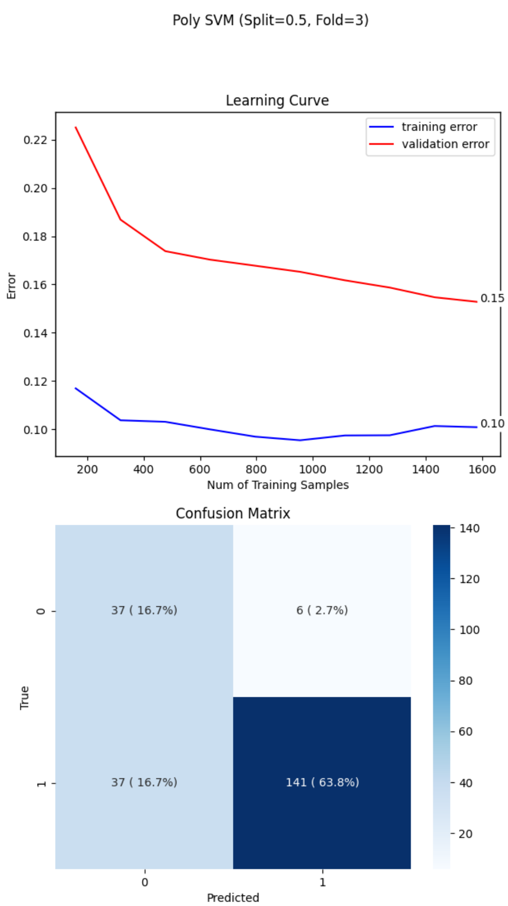
L. SVM-Poly Kernel

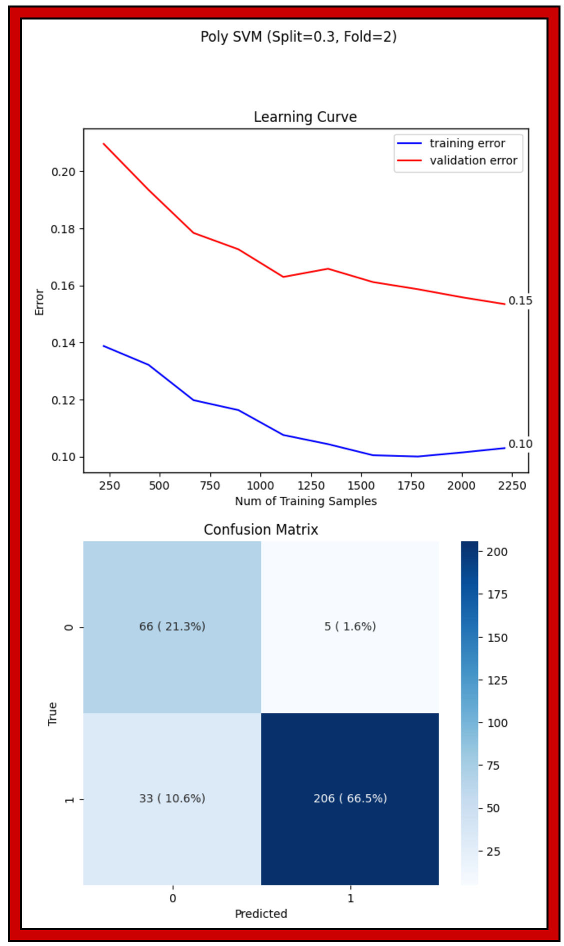
50/50 Split

	Precision	Recall	F1score	Accuracy
0	0.878	0.874	0.866	0.874
1	0.856	0.833	0.820	0.833
2	0.815	0.805	0.789	0.805
3	0.889	0.887	0.883	0.887
4	0.874	0.851	0.841	0.851
5	0.827	0.819	0.809	0.819
6	0.868	0.869	0.863	0.869
7	0.872	0.855	0.846	0.855
8	0.878	0.873	0.866	0.873
9	0.823	0.828	0.824	0.828

70/30 Split

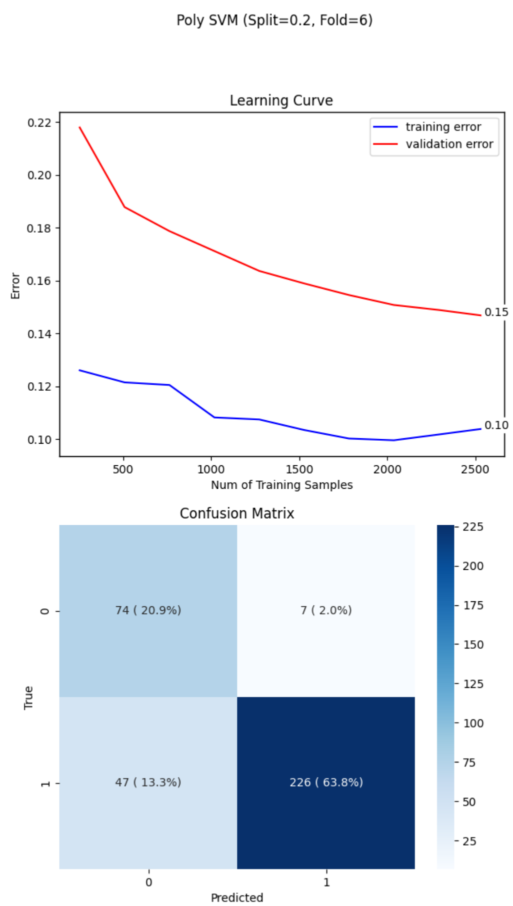
	Precision	Recall	F1score	Accuracy
0	0.846	0.835	0.822	0.835
1	0.884	0.877	0.871	0.877
2	0.893	0.890	0.886	0.890
3	0.858	0.852	0.841	0.852
4	0.867	0.868	0.863	0.868
5	0.825	0.816	0.804	0.816
6	0.851	0.851	0.845	0.851
7	0.884	0.883	0.881	0.883
8	0.863	0.858	0.850	0.858
9	0.827	0.822	0.813	0.822





80/20 Split

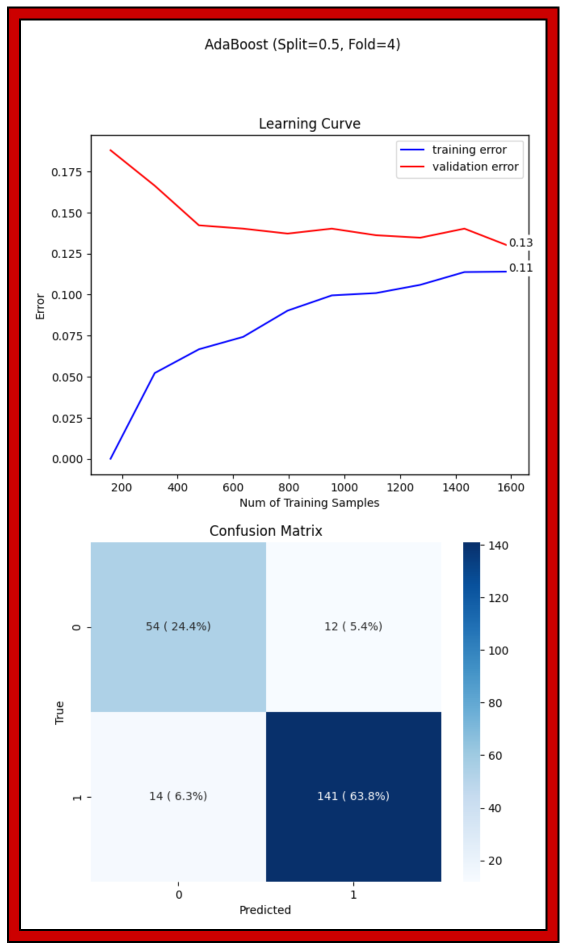
	Precision	Recall	F1score	Accuracy
0	0.844	0.842	0.831	0.842
1	0.876	0.862	0.851	0.862
2	0.847	0.847	0.840	0.847
3	0.857	0.859	0.853	0.859
4	0.859	0.847	0.840	0.847
5	0.857	0.847	0.838	0.847
6	0.894	0.890	0.886	0.890
7	0.869	0.867	0.863	0.867
8	0.856	0.847	0.840	0.847
9	0.848	0.841	0.830	0.841



M. *AdaBoost Classifier*

50/50 Split

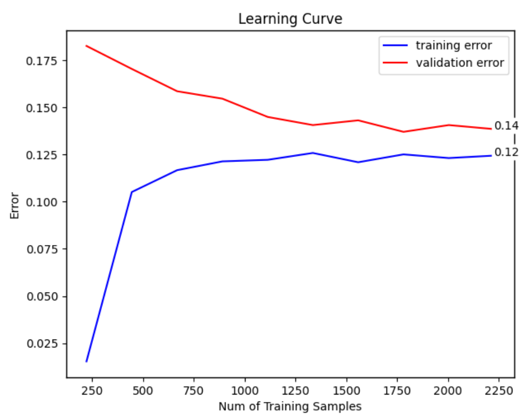
	Precision	Recall	F1score	Accuracy
0	0.897	0.896	0.893	0.896
1	0.875	0.874	0.871	0.874
2	0.886	0.887	0.885	0.887
3	0.882	0.882	0.882	0.882
4	0.895	0.891	0.889	0.891
5	0.852	0.851	0.847	0.851
6	0.845	0.846	0.845	0.846
7	0.855	0.855	0.852	0.855
8	0.871	0.873	0.872	0.873
9	0.854	0.855	0.854	0.855

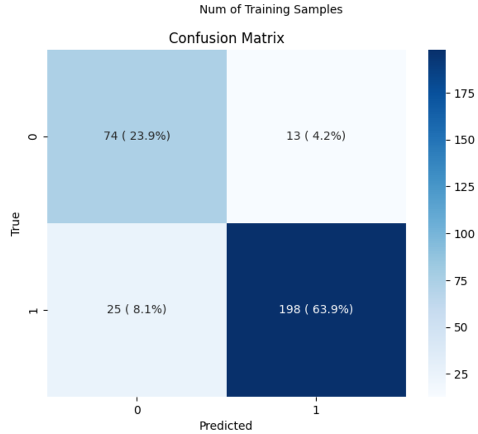


70/30 Split

	Precision	Recall	F1score	Accuracy
0	0.859	0.861	0.858	0.861
1	0.876	0.877	0.875	0.877
2	0.916	0.916	0.915	0.916
3	0.847	0.848	0.842	0.848
4	0.862	0.865	0.862	0.865
5	0.851	0.852	0.849	0.852
6	0.832	0.835	0.832	0.835
7	0.876	0.874	0.874	0.874
8	0.852	0.854	0.852	0.854
9	0.857	0.858	0.855	0.858

AdaBoost (Split=0.3, Fold=2)

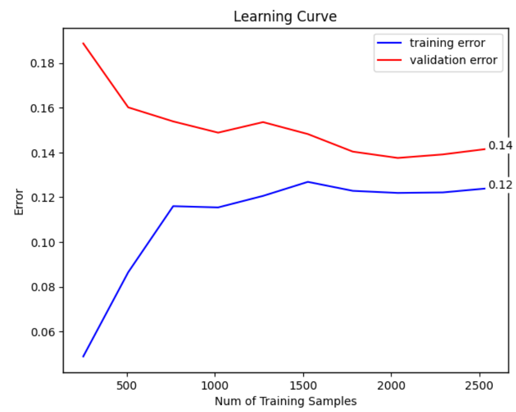




80/20 Split

	Precision	Recall	F1score	Accuracy
0	0.850	0.853	0.850	0.853
1	0.874	0.873	0.868	0.873
2	0.874	0.876	0.874	0.876
3	0.865	0.867	0.864	0.867
4	0.864	0.864	0.862	0.864
5	0.871	0.870	0.866	0.870
6	0.883	0.884	0.882	0.884
7	0.873	0.873	0.873	0.873
8	0.850	0.850	0.847	0.850
9	0.826	0.830	0.826	0.830

AdaBoost (Split=0.2, Fold=6)



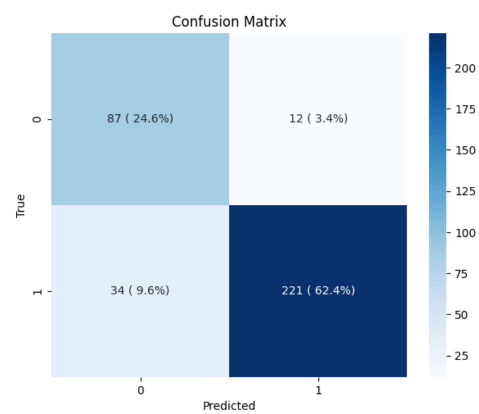
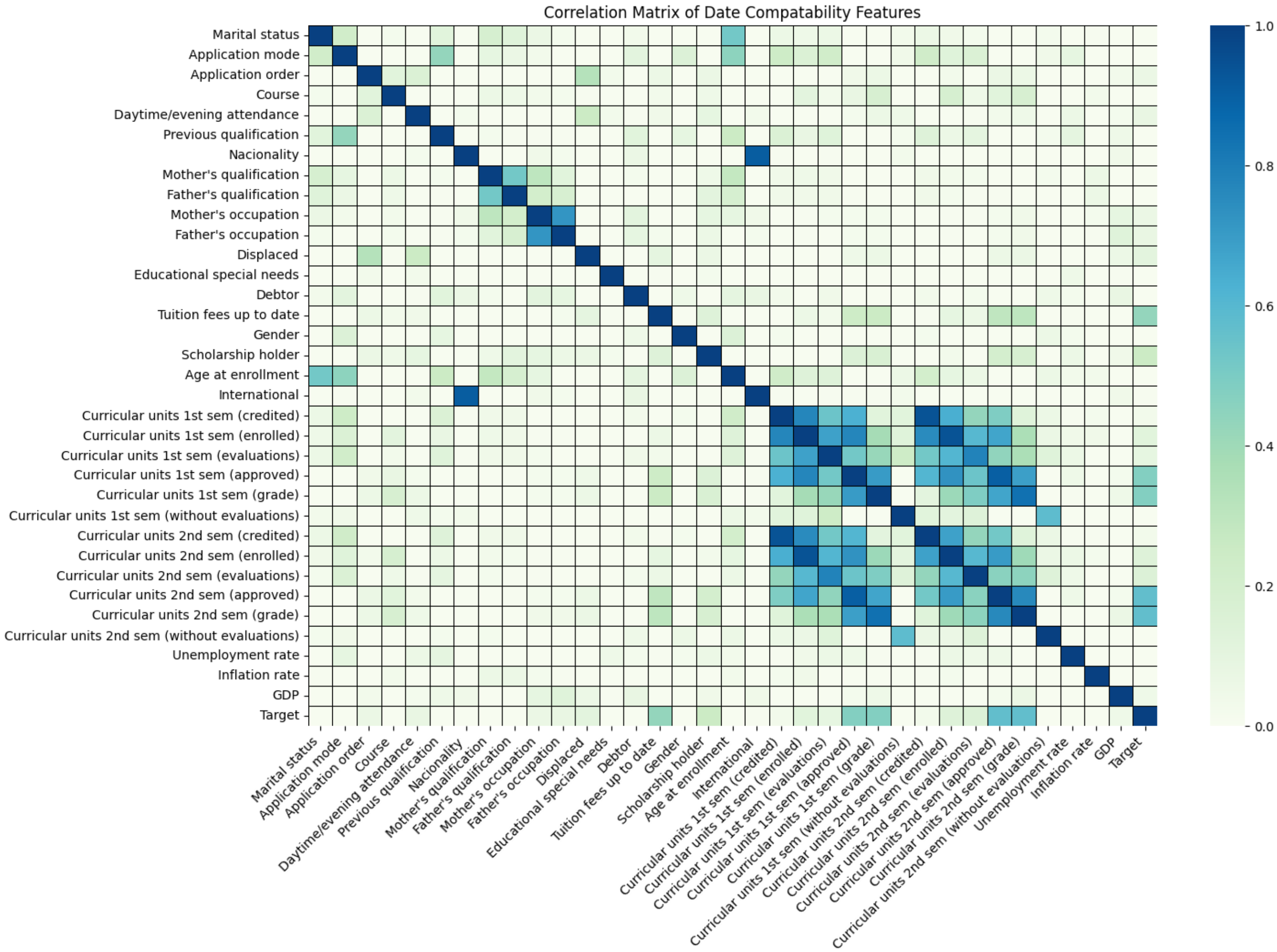


Figure 1. Correlation Matrix of all Features



VII. Feature Analysis

One thing that was not done in this project was dimensionality reduction. We decided to do some feature analysis during the post analysis review to get a deeper understanding of the dataset and some possible insight to why some models behaved how they did. Figure 1 above shows a complete correlation matrix of all the features. From this matrix we can quickly see that the majority of features are not correlated to each other. There is a section of relatively correlated features towards the bottom right; these features essentially describe the amount of college credits each student took and got credit for per semester. The correlation is seen between the features from semester 1 and semester 2. Though the plot shows correlation here, it is due to the nature of these features and so it doesn't hold much statistical significance.

Additionally, we used principal component analysis to visualize how we could have reduced some dimensionality in our dataset. In Figure 2A and 2B below, we calculated and tabulated the PCA variance ratio for each feature and the total sum of PCA variance.

The PCA variance ratio shows us the proportion of the total variance in the dataset that is explained by each principal component. This is highly valuable since we can compare how much of the information in the dataset is captured by each principal component.

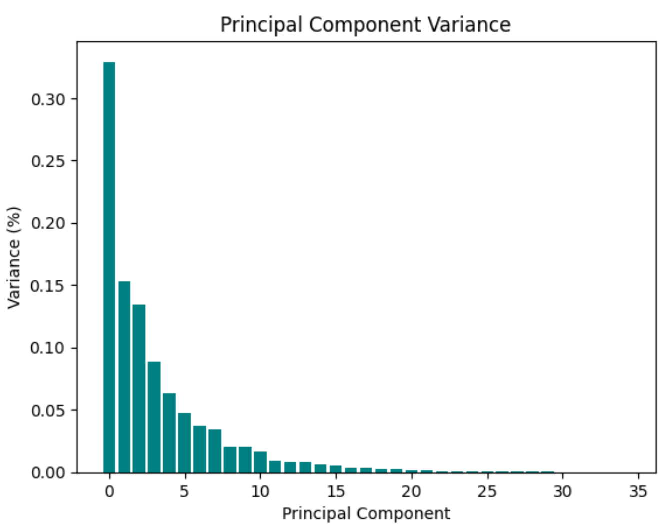
The feature with the highest PCA variance ratio was 'Marital Status' with .329. The next highest was 'Application Mode' with .153, so it is clear that 'Marital Status' explains the most variance by far. Though our dataset has 35 features, just the top nine features account for 90% of the variance and the top 12 account for 95%.

This signifies there is definitely room for dimensionality reduction with this dataset. This is a major area for improvement in this analysis is definitely something we would do if we were to redo this analysis.

Figure 2A. Table of PCA Variance

	Feature	PCA Variance Ratio	Sum PCA Variance
0	Marital status	0.329053	0.329053
1	Application mode	0.153099	0.482152
2	Application order	0.134038	0.616190
3	Course	0.088379	0.704570
4	Daytime/evening attendance	0.063206	0.767776
5	Previous qualification	0.047283	0.815058
6	Nacionality	0.036950	0.852009
7	Mother's qualification	0.033887	0.885896
8	Father's qualification	0.020682	0.906578
9	Mother's occupation	0.020508	0.927086
10	Father's occupation	0.016302	0.943388
11	Displaced	0.009403	0.952791
12	Educational special needs	0.008293	0.961084
13	Debtor	0.007700	0.968784
14	Tuition fees up to date	0.006287	0.975071
15	Gender	0.005717	0.980788
16	Scholarship holder	0.003898	0.984686
17	Age at enrollment	0.003668	0.988354
18	International	0.002777	0.991131
19	Curricular units 1st sem (credited)	0.002655	0.993786
20	Curricular units 1st sem (enrolled)	0.001484	0.995270
21	Curricular units 1st sem (evaluations)	0.001321	0.996591
22	Curricular units 1st sem (approved)	0.000539	0.997130
23	Curricular units 1st sem (grade)	0.000451	0.997580
24	Curricular units 1st sem (without evaluations)	0.000426	0.998006
25	Curricular units 2nd sem (credited)	0.000382	0.998388
26	Curricular units 2nd sem (enrolled)	0.000352	0.998741
27	Curricular units 2nd sem (evaluations)	0.000310	0.999051
28	Curricular units 2nd sem (approved)	0.000282	0.999332
29	Curricular units 2nd sem (grade)	0.000222	0.999555
30	Curricular units 2nd sem (without evaluations)	0.000170	0.999724
31	Unemployment rate	0.000136	0.999860
32	Inflation rate	0.000110	0.999970
33	GDP	0.000022	0.999992
34	Target	0.000008	1.000000

Figure 2B. PCA Variance per ID of Feature



VIII. Conclusion

A. Overall Performance Metrics

50/50 Split

Model	Best Fold	Precision	Recall	Accuracy
Decision Tree	1	.841	.842	.842
Random Forest	3	.909	.910	.910
Perceptron	3	.871	.873	.873
Guassian NB	3	.869	.855	.855
Logistic Regression	3	.913	.914	.914
Linear SVM	0	.901	.901	.901
RBF SVM	8	.898	.896	.896
Gradient Boosting	3	.890	.891	.891
Multi Layer Perceptron	3	.898	.896	.896
K Nearest Neighbor	3	.863	.864	.864
Poly SVM	3	.889	.887	.887
AdaBoost	4	.895	.891	.891

50/50 Split

Model		RMSE	r2
Linear Regression	3	0.276	0.643

70/30 Split

Model	Best Fold	Precision	Recall	Accuracy
Decision Tree	2	.830	.819	.819
Random Forest	2	.902	.903	.903
Perceptron	1	.876	.874	.874
Guassian NB	2	.870	.871	.871
Logistic Regression	2	.912	.913	.913
Linear SVM	2	.923	.923	.923
RBF SVM	2	.914	.913	.913
Gradient Boosting	2	.922	.923	.923
Multi Layer Perceptron	2	.919	.919	.919
K Nearest Neighbor	2	.882	.881	.881
Poly SVM	2	.893	.890	.886
AdaBoost	2	.916	.916	.916

70/30 Split

Model		RMSE	r2
Linear Regression	2	0.282	0.633

80/20Split

Model	Best Fold	Precision	Recall	Accuracy
Decision Tree	7	.824	.819	.819
Random Forest	1	.899	.895	.895
Perceptron	6	.872	.873	.873
Guassian NB	6	.850	.850	.850
Logistic Regression	2	.888	.890	.890
Linear SVM	6	.887	.887	.887
RBF SVM	1	.892	.884	.884
Gradient Boosting	6	.901	.898	.898
Multi Layer Perceptron	6	.889	.890	.890
K Nearest Neighbor	3	.842	.845	.845
Poly SVM	6	.894	.890	.890
AdaBoost	6	.883	.884	.884

80/20Split

Model		RMSE	r2
Linear Regression	6	0.305	0.578

B. Best Fold and Splits

After outputting all the performance metrics for each fold on each split for each model, this is a summary of the best fold and split or each model:

Best Fold per Split

Model	50/50	70/30	80/20	Best Split
Decision Tree	1	2	7	50/50
Random Forest	3	2	1	50/50
Perceptron	3	1	6	50/50
Naive Bayes	3	2	6	50/50
Logistic Regression	3	2	2	50/50
Linear Regression	3	2	6	50/50
SVM - Linear	0	2	6	70/30
SVM - RBF	8	2	1	50/50
Gradient Boosting	3	2	6	80/20
MLP	3	2	6	80/20
KNN	3	2	3	50/50
SVM - Poly	3	2	6	70/30
AdaBoost	4	2	6	50/50

**notes: fold numbers start at 0*

Split	Best Fold (by model)	Frequency as Best Split
50/50	3	9
70/30	2	2
80/20	6	2

The most interesting pattern here is how the 50/50 split was by far the most common split. This indicates that a balanced split of the data was generally more effective for predicting the target variable. This could be due to the fact that a balanced split ensures that both the training and testing sets have a similar distribution of the target variable, allowing the model to better generalize to new data.

Only the SVM-Linear model had the 70/30 split as its best. Our theory is that since the linear kernel is a simple algorithm that tries to fit a linear relationship, having more training data than the 50/50 split but less than the 80/20 split allowed it to capture the underlying patterns in the data the best.

It is also interesting to see that the Multi-Layer Perceptron model's best split was 80/20. This model is generally more complex and requires a large amount of data to train effectively, so it is possible that the 80/20 split was just the best split due to the nature of the MLP model.

The most common fold for splits 50/50, 70/30, 80/20 were 3,2,6 respectively. Interestingly, for the 70/30 split fold 2 was the best fold for every model but 1. Our hypothesis for one fold being dominant in each split is that since our k-fold cross validation had the shuffle parameter set to true, the data was randomly shuffled and split into fold which could have led to certain folds having a more representative sample of the overall data than others. This difference in representativeness is magnified when you have three splits since different splits will produce different distributions of data for the folds to generate from. Combining this with our aforementioned imbalance class leads to the possible explanation that certain folds may just be performing better due to better distributions of the target.

C. The Best Model

To determine the best model we looked at the confusion matrix ratios, f1-scores, and learning curves. Our top 2 models were Naive Bayes and Logistic Regression with both on fold 3 in the 50/50 split. The following figure is a summary of basic performance metrics for each model:

Gaussian NB (50-50 Split, Fold 3)

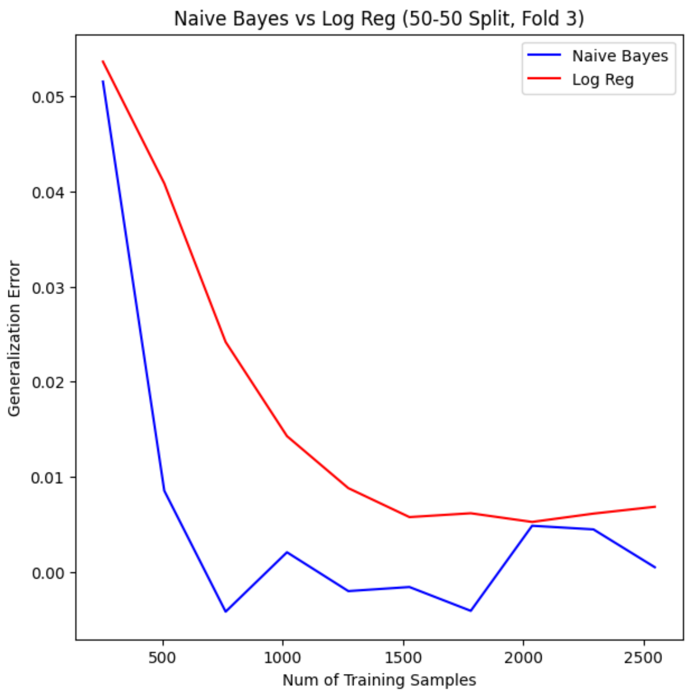
	Precision	Recall	F1score	Accuracy
3	0.869	0.855	0.858	0.855

Logistic Regression (50-50 Split, Fold 3)

	Precision	Recall	F1score	Accuracy
3	0.913	0.914	0.913	0.914

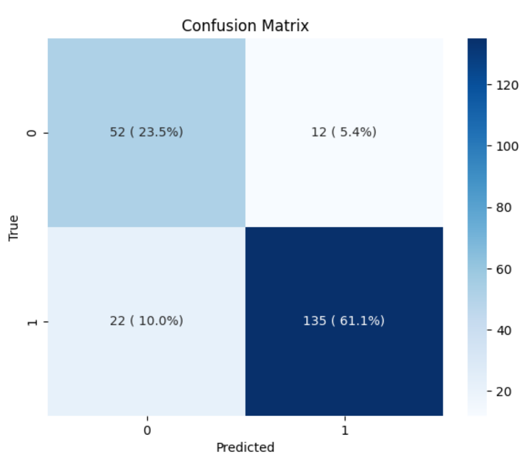
The logistic regression model has higher precision, recall, and F1-score than the Naive Bayes model. These metrics are particularly important for a binary classification problem where the two classes are imbalanced, as in this case where the 0 class is much less frequent than the 1 class. A high precision means that the model has a low false positive rate, which is important for minimizing the number of false positive predictions. A high recall means that the model has a low false negative rate, which is important for ensuring that all relevant positive cases are identified. The F1-score is the harmonic mean of precision and recall, and provides a balanced measure of the two metrics.

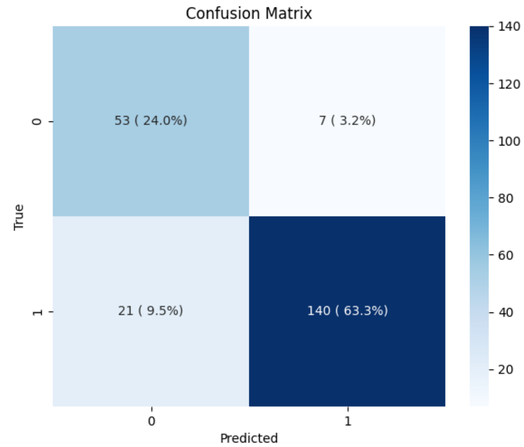
The following figure compares the generalization error for both models:



Both models have a very low generalization error, indicating that they are less prone to overfitting and will behave relatively well with new data. Though the Naive Bayes model is slightly better than Logistic Regression the degree of magnitude is .001, and is effectively negligible. The curve of Logistic Regression is smoother than that of the Naive Bayes model, this means that it is more consistent as the number of training samples increases. We chose to consider above the actual generalization error value, which results in the Logistic Regression also performing better in this category. We then looked at their confusion matrices:

Gaussian Naive Bayes





The logistic regression model has a higher true positive rate and a lower false positive rate than the Naive Bayes model. This means that the logistic regression model is better at correctly identifying positive cases and has fewer false positives, which is important in this binary classification problem.

In summary, the logistic regression model is better than the Naive Bayes model for this binary classification problem, as it has higher precision, recall, F1-score, accuracy, and is less prone to overfitting.

D. Post Analysis Review and Thoughts

We have a couple ideas as to why the logistic regression model was the best performing model. The biggest reason we hypothesized is that our dataset has an imbalanced class distribution and general logistic regression works well when on these types of datasets that are also a binary classification. This is because of the nature of the sigmoid function and the model is able to adjust the decision threshold to better classify the minority class. Naive Bayes, on the other hand, assumes that features are independent and this can be sensitive to imbalanced data. Our dataset also has no continuous features which possibly let the logistic regression model have an easier time establishing a linear correlation between the features and target.

Neither this model nor this analysis is perfect however. A big room for improvement is to see if dealing with the imbalanced class distribution alters the models' performance. This could lead to a different best model than logistic regression. One way to deal with imbalanced class distribution is to use sampling techniques, such as oversampling or undersampling, to balance the class distribution. Oversampling involves increasing the number of instances in the minority class, while undersampling involves reducing the number of instances in the majority class. These techniques can help prevent models from being biased towards the majority class. Another way to deal with imbalanced class distribution is to use class weighting during model training.

Class weighting assigns a higher weight to the minority class during training, which can help to improve the model's performance on the minority class.

Another area of improvement for this analysis would be hyper-parameter optimization. While the default settings yielded good results, tuning the parameters could result in a different best model and change the analysis conclusion.

Testing the models across three different splits and using a 10-fold cross validation was an extremely important aspect of this analysis. This let us get a better idea of how each model behaved with the dataset and overall led to a better final model. Nevertheless the two aforementioned areas of improvement would be opportunities that we would seriously consider if we were to redo this analysis.

Ultimately, it seems our processed dataset and default model settings were still able to produce solid performance metrics. Logistic regression was the best model with this data and its generated metrics signified it be the best at handling future data as well.

IX. References

[1] Alsariera YA, Baashar Y, Alkawsi G, Mustafa A, Alkahtani AA, Ali N. Assessment and Evaluation of Different Machine Learning Algorithms for Predicting Student Performance. *Computational Intelligence & Neuroscience*. May 2022:1-11. doi:10.1155/2022/4151487

[2] Baig, M. A., Shaikh, S. A., Khatri, K. K., Shaikh, M. A., Khan, M. Z. ., & Mahira Abdul Rauf. (2023). Prediction of Students Performance Level Using Integrated Approach of ML Algorithms . *International Journal of Emerging Technologies in Learning (iJET)*, *18*(01), pp. 216–234. https://doi.org/10.3991/ijet.v18i01.35339

[3] Valentim Realinho, Jorge Machado, Luís Baptista, & Mónica V. Martins. (2021). Predict students' dropout and academic success (1.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5777340