Project 2: Supervised Learning

Building a Student Intervention System

1. Classification vs Regression

Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?

> This is a classifcation problem, where the goal is to predict student performance outcomes with binary output -- whether the student passed ("yes"), or didn't pass ("no"). Classification is for solving problems with discrete value outputs.

2. Exploring the Data

- Total number of students = **395**
- Number of students who passed = **265**
- Number of students who failed = **130**
- Graduation rate of the class (%) = **67.09**
- Number of features = **30**

3. Preparing the Data

- Identify feature and target columns
- Preprocess feature columns
- Split data into training and test sets

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem.

Produce a table showing training time, prediction time, F_1 score on training set and F_1 score on test set, for each training set size.

Model #1: Logistic Regression

- What are the general applications of this model? What are its strengths and weaknesses?

  > **Logistic regression** builds a linear model for classification and can find a linear separator for a dataset with a binary target variable, such as the student data "passed" label.
  >
  > Strengths: Fast, simple. Gives insight into the impact of each feature. Less prone to overfitting (higher bias, lower variance). By generating a linear decision boundary, it combats the curse of dimensionality, where the more features a dataset has, the more data is needed to model it.
  >
  > Weaknesses: By assuming that a linear decision boundary exists, it may be less accurate than other models and be prone to underfitting.

- Given what you know about the data so far, why did you choose this model to apply?

  > With limited data and a relatively large feature set, it makes sense to keep the model simple and avoid overfitting. The model can also provide a probability of a given label prediction being true.

Logistic regression	Training set size 100	Training set size 200	Training set size 300
Training time (secs)	0.002	0.002	0.005
Prediction time (secs)	0.000	0.000	0.000
F_1 score for training set	0.8571	0.8380	0.8381
F_1 score for test set	0.7612	0.7794	0.7910

Model #2: SVM (Support Vector Machine)

- What are the general applications of this model? What are its strengths and weaknesses?

> **Support Vector Machines** can find a hyperplane to separate data within a dataset to perform a classification task. The separator is found by using support vectors to maximize the margins around the decision boundary.
>
> Strengths: Works well with high dimension data and can still be effective where the number of dimensions is greater than the number of samples. Has flexibility to specify the kernel function for the decision function, which enables transforming the data into a higher dimensional space in order to find a separator.
>
> Weaknesses: Doesn't directly provide a probability estimate for predictions. The kernel function can be prone to overfitting.

- Given what you know about the data so far, why did you choose this model to apply?

> With limited data and a relatively large feature set, it makes sense to use the kernel trick with the SVM to achieve higher prediction accuracy.

SVM	Training set size 100	Training set size 200	Training set size 300
Training time (secs)	0.004	0.005	0.011
Prediction time (secs)	0.002	0.003	0.006
F_1 score for training set	0.8591	0.8693	0.8692
F_1 score for test set	0.7838	0.7755	0.7586

Model #3: Nearest Neighbor

- What are the general applications of this model? What are its strengths and weaknesses?

> **Nearest Neighbor** methods can perform classification via instance-based learning, where instances of training data are stored and a predefined number of training instances closest in distance to new test data are used to predict the test label.
>
> Strengths: Simple and fast to train. Keeps all data instead of discarding after model-creation. Being non-parametric makes it effective in finding irregular decision boundaries.
>
> Weaknesses: Slower to query predictions. Needs to store data to make prediction. All features are considered equally.

- Given what you know about the data so far, why did you choose this model to apply?

> With limited data, it makes sense to keep as much data as possible to make predictions rather than generate a model that overfits. The model can also provide a probability of a given label prediction being true.

KNN	Training set size 100	Training set size 200	Training set size 300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.023	0.003	0.008
F_1 score for training set	0.7972	0.8571	0.8722
F_1 score for test set	0.7068	0.7121	0.7482

5. Choosing the Best Model

- Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model. Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?

> A Logistic Regression model was chosen as the best model. It provides the best balance of model simplicity and prediction accuracy, which should limit computer resource needs. During experiments on the data using different models, it had the lowest combo of computation time for both training the computer to model the student data as well as make predictions on test cases (it took .005 sec compared to .017 and .009 for the others). The Logistic Regression model also had the highest score on experimental tests, achieving an F_1 score of 0.7910 vs 0.7586 and 0.7482 for the others (note: F_1 measures performance of the model by looking at both the accuracy of each prediction whether a student passes, as well as breadth in identifying all the students who actually passed).
>
> The simplicity of the model should prevent it from suffering from overfitting, which will allow it to predict well on new students who aren't in the current data. The model can be learned on limited amounts of data, which we have with the student data, and gives a probability of a prediction being correct in addition to the "pass/didn't pass" prediction itself. In addition, the model can provide insight into the specific attributes of the data that contribute greater weight to the prediction, allowing for a focus on students who display similar attributes.

- In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work.

> Using the student data provided, a model has been built that gives a weight to student attributes found within the data, which are referred to as "features." For example, the data contains features about each student such as their age, their parents' occupation, and the amount of time spent studying. The weights in the constructed model correspond to the importance of each feature in the data when predicting whether a student is likely to pass or not -- the model essentially provides an equation which can be used to calculate the likelihood that a new student will pass, and then a decision can be made on whether an intervention is appropriate.
>
> The model simply needs the features of a new student to plug into the equation and return a probability (0-100%) that the student will pass or not. The model will predict a "yes" for students with a probability of greater than 50%, and predict "no" otherwise.

- Fine-tune the model. Use Gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.

> Used Gridsearch to tune the parameters C, class_weight, and max_iter using the following settings:

```
parameters = {'C': (1.0, 0.5, 1.5),
              'class_weight': (None, 'balanced'),
              'max_iter': (100, 200, 50)}
```

> The best estimator was:

```
LogisticRegression(C=0.5, class_weight=None, dual=False, fit_intercept=True,
      intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
      penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
      verbose=0, warm_start=False)
```

- What is the model's final F_1 score?

> Final F_1 score is 0.8.