Course Project Phase 2
Robbie & Parsa

Binary Dataset: ISOT Fake News Detection Dataset

For the binary classification dataset, the Fake and Real News Dataset, the three models we will be testing are logistic regression, support vector machine, and multinomial Naive Bayes. Each of these models perform well on text analysis and are each likely to provide models that can correctly predict test data.

Logistic regression is fast to train and predict and it is easy to regularize, which will be beneficial because of the large size of the dataset. It performs well on highly dimensional data such as news articles and outputs probabilities, which could give us more insight into the overall strength of the model. The main hyperparameters we will be tuning are the regularization strength and type, so adjustments can be made to the hyperplane if overfitting occurs, and the learning rate, to ensure that we don't converge too slowly to be able learn the dataset or too quickly such that we learn too simplistic of a model. The dataset will be transformed by the Scikit-learn TF-IDF vectorizer so that it can be interpreted by the model. The TF-IDF hyperparameter ngram_range will be tuned to control how many words are considered during vectorization.

The second model we will be training is support vector machines. Like logistic regression, they are also good at learning highly dimensional data and are easy to regularize. One benefit of SVM over logistic regression is that its kernel functions can be used to learn non-linear decision boundaries, which may be helpful since we will not know if the data is linearly separable or not until after it is put through the vectorization function. The hyperparameters that will be tuned for SVM include slack regularization, so that the model can handle outliers or noise that may occur in the dataset, as well as kernel type, since one method may perform better on this dataset versus the others. Like the logistic regression model, the dataset will need to be vectorized by TF-IDF so that the SVM can learn it.

The final model we will test is multinomial Naive Bayes. It outputs probabilities, which will be helpful in determining the strength of the model, and it is computationally efficient, which will be beneficial for the large dataset. Unlike logistic regression and SVM, it is naturally able to process text and therefore will not need the dataset to be transformed using TF-IDF. The hyperparameters we will tune are the smoothing factor, which may have some effect on the probabilities, and the class priors, which will affect how the prior probabilities are calculated.

The two main metrics we will use to measure the strength of these models are accuracy and recall. Since the dataset is balanced, we do not need to worry about the downsides that come with using accuracy. It should give a general idea of how well each model performed. We will also be using recall over precision because we would rather false positives over false negatives. In other words, it is better to label a real news article as fake rather than label a fake article as real. These metrics will allow us to compare the performance of each model against the rest and decide which model is the best for this dataset.

Multiclass Classification Dataset: The Customer Segmentation Dataset

The three models we will train on our multiclass classification dataset, the Customer Segmentation Dataset, will be decision tree, k-nearest neighbors, and artificial neural network. Each of these models are able to learn non-linear decision boundaries and can handle multiple classes.

The first model we will train is the decision tree. Decision trees are simple to implement and work well with the feature types in our dataset. It also ignores irrelevant features, which will keep the model from making decisions based on features that don't affect the truth labels. The hyperparameters we will tune are the max depth, which will control how complex the model is and will prevent overfitting, and the splitting criteria, either if the tree splits on the best feature or a random one.

The second model we will train is k-nearest neighbors. Like decision trees, KNN is a simple model that works very well with the types of features in the dataset, however it isn't as prone to overfitting as decision trees are. The hyperparameters we will tune are the number of neighbors, which will affect the results and will control overfitting. We will also tune the normalization functions between min-max and z-score, since we will want our data to be scaled properly and each normalization function may have varying results. Something to be concerned about with KNN is the curse of dimensionality. This should not be a problem since this dataset does not contain noticeably highly dimensional data, however we should monitor this during training to ensure it does not cause us to come to incorrect conclusions.

The final model we will be training is an Artificial Neural Network. Whereas decision trees and KNN are simpler models, ANN is a more complex model which could give us a more accurate result that doesn't have the same overfitting concerns that decision trees or KNN has. Although overfitting can still occur, it shouldn't be as much of an issue. The hyperparameters we will tune are the number and size of hidden layers, which will affect the complexity of the model and may keep the model from being unable to learn the data. We will also tune the learning rate, which will help to tune the model to be able to learn the optimal model, as well as the activation functions, which may have an effect on the probabilities that are calculated. One thing to be concerned about is not having a large enough sample such that the model can't learn the dataset. ANN requires a large dataset in order to properly learn a decision boundary, so if our dataset turns out to not be sufficient, this may be an issue. However, tuning the number of hidden layers and how many nodes are in each hidden layer should help to increase complexity.

The two metrics we will use to measure each model are accuracy and F1 score. It is not clear if the dataset is balanced or not, so we will not know if it will be until we start training. Accuracy provides a quick and easy way to measure the strength of a model, but may be misleading if the dataset is unbalanced. F1 score provides a more reliable score for datasets that are unbalanced, and since we do not prefer either false positives or negatives, we can use F1 to balance precision and recall which will give us a more general idea of the strength of each model.