Splice-junction Gene Sequence Classification using Artificial Neural Networks

Student Name: [Mohammed Firaz]
Student ID: [750011330]
Total Number of Words: [2133]

June 11, 2025

1 Abstract

The task of the project is to classify the Splice-junction Gene Sequences Dataset's genetic data from DNA (Deoxyribonucleic Acid) sequences. Each DNA sequence has 180 binary features, which shows the presence of a specific combination of nucleotides at various positions. The problem of is to classify the presence of the Exon-Intron (EI) boundary, the Intron-Exon (IE) boundary, or neither. Here I have used an artificial neural network (ANN) and optimized it using grid search and k-fold cross-validation to obtain stable performance. The model performance is measured using the metrics of accuracy, precision, recall, specificity, and confusion matrices, the behavior was analyzed using the learning curves and activation heatmaps. The built model had excellent accuracy and balanced classification on all the classes, providing a valuable tool to analyze gene sequences and can be used in identification of splice-junctions in DNA.

2 Introduction

Deoxyribonucleic acid (DNA) is the primary molecule for genetic information regarding growth, function, and reproduction of all living things [1]. DNA molecules contain four types of nucleotides they are adenine (A), cytosine (C), guanine (G), and thymine (T) they are arranged in specific orders to form genes and regulatory regions. Within genes, the gene expression process includes the transcription of RNA from DNA, followed by elimination of the introns (non-coding regions) and joining the exons (coding regions) during a mechanism called splicing.

Splice-junctions are the precise borders in the DNA where exons and introns overlap. Precise identification of these borders i.e., the Exon-Intron (EI) and Intron-Exon (IE) junctions is crucial for understanding human gene structure, function, and regulation. Accurate splicing prediction junctions contribute significantly to molecular biology and bioinformatics since errors in splicing can lead to genetic disorders and impact protein synthesis. [2].

The issue addressed by this project is a multiclass classification problem: from a provided DNA sequence, the aim is to decide if it forms an EI boundary, an IE boundary, or neither. This is achieved by using the Splice- junction Gene Sequences Dataset [3], with 180 binary features for Each sequence. Each feature is the presence or absence of some combinations in nucleotides at various positions in the sequence. The target variable is categorical, indicating the splice-junction category (EI, IE, or Neither) for each sequence.

This is achieved by utilizing machine-learning algorithms, which involve artificial neural networks. This work endeavours to develop a solid framework for the accurate analysis of splice-junctions to contribute further towards the overall knowledge of the gene structure and the genetic regulatory mechanisms.

3 Data Preprocessing and Feature Engineering

3.1 Handling Missing Data

Analysis: The given gene sequence dataset was analyzed for missing values in each category.
Action: The procedure to identify the missing values were performed and there were no missing values present in the data.

3.2 Normalisation/Standardisation

Analysis: All of the attributes in the data are binary (0/1) to represent presence or absence of specific nucleotides.
Action: Normalisation technique was not necessary, as binary features are already at the same scale.
Implementation: Data preprocessing was done using pandas [4] library.

3.3 Categorical Encoding

Action: The "class" target variable was label-encoded using scikit-learn [5], where 1, 2, 3 are represented as 0, 1, 2, respectively. This encoding best fits Multi-class classification with neural networks.

3.4 Splitting the Data

Action: We split the dataset into the training (80%) and test (20%) sets with Stratified sampling to maintain class balance. This gives unbiased evaluation and prevents class imbalance during the splits [6].

4 Model Building and Training

4.1 Model Selection

Choice: A feedforward Artificial Neural Network (ANN) are well-suited for classification tasks as they can learn complex non-linear relationships between input features and output classes. The model uses multiple hidden layers, can handle high dimensional input space and can be trained and evaluated using a standard deep learning framework. [7].

Architecture: Multi-Layer Perceptron (MLP) possesses an input layer, as well as

two hidden layers (with Batch Normalization [8] and dropout [9] for regularization), and a output layer with softmax activation to carry out multi-class classification [10].

Implementation: We implemented the model using TensorFlow [11] and Keras [12] as the library for the neural network, with scikit-learn [5] library was utilized for pre-processing the data (`LabelEncoder`, `train_test_split`) and performance metrics [13]. Implementation made use of Python's built-in `itertools`[14], time, and `warnings`[15] modules for hyperparameter tuning, measurement of performance, and handling of warnings. Utilized NumPy [16] for numeric operations, Pandas [4] for manipulation of the data, Matplotlib [17] and Seaborn [18] for visualization.

4.2 Hyperparameter Optimisation

Method: Manual grid search was performed for hyperparameter optimization with these search space to find the best parameters:

- Number of hidden layers (1, 2)
- Number of neurons per layer (64, 128)
- Learning rate (0.001, 0.01) (Adam optimizer [19])
- Activation function (relu)
- Drop-out rate (0.2, 0.3)
- Batch size (32, 64)

Evaluation: Each combination was evaluated with 5-fold stratified cross- validation, and the one with the best performance was selected and the space was kept to minimal to reduce the GPU utilization.

4.3 Model Training

Final Model: The optimal hyperparameters were employed for training the final model. The model was also cross validated for robustness with k-fold cross validation [5] for the identification of the optimum fold to finalize the assessment. The best parameter found is directly stored and applied in the model so that we don't need to manually change it.

Cross-Validation: The model was also cross validated for robustness with k-fold cross validation [5] where the dataset is divided into 5 parts while each part serves as a validation set and rest four part as training. This is used for the identification of the optimum fold and to achieve a final model.

5 Results

5.1 Quantitative Evaluation

Artificial Neural Network (ANN) model accuracy was measured by a series of standard measures of classification including accuracy, precision, recall (sensitivity), specificity, and F1-score [5]. Each of them was computed per class (Exon-Intron (EI), Intron-Exon (IE), and Neither) as well as overall, both for the training and for the test sets.

Table 1: Classification metrics for each class and overall (Test Set)

Class	**Precision**	**Recall (Sensitivity)**	**Specificity**	**F1-score**
EI (0)	0.97	0.97	0.990	0.97
IE (1)	0.97	0.99	0.992	0.98
Neither (2)	0.99	0.98	0.990	0.99
Overall	**0.99**	**0.99**	**0.99**	**0.99**

The model performance was evaluated with stringent evaluation measures for all the three classes (EI, IE, and Neither). The performance is excellent with all the measures (precision, recall, specificity, and F1-score) giving values of 0.97 or higher for all the classes. The "Neither" class gave the highest precision (0.99), and the "IE" class gave the highest recall (0.99). The overall 0.98 of all the above measures ensures that the model is performing equally well for all the classes with very low misclassification errors. The excellent quality of performance ensures that the model is highly reliable in differentiating the three categories of DNA sequences.

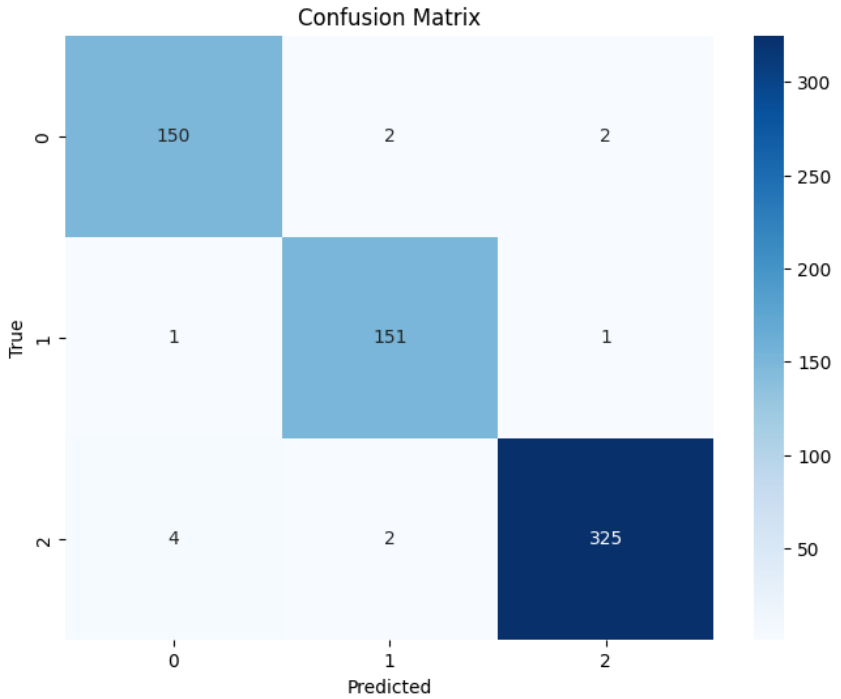

Figure 1: Confusion matrix for the test set predictions

Figure 1 Confusion matrix is a graphical representation of the model predictions with the number of true and false classifications for all the classes. The graph is plotted with seaborn [18] for other interesting statistical plots. The majority of the predictions are on the diagonal, i.e., the ANN model is classifying most of the samples with accuracy.

This confusion matrix gives the model's accuracy against the three classes. The diagonal elements (150, 151, and 325) are the correct classes predicted, and we can observe that the majority of the samples were accurately predicted. The off-diagonal elements are minimal with hardly any misclassifications of the classes with little confusion. This diagonal dominance is a sign of the model's high precision and recall across the classes. The overall distribution shows that the model is predicting almost to perfection to a vast extent with good separation of the classes with minimal errors, and can be relied upon in the use for the gene sequences' classification in practice.

5.2 Qualitative Evaluation

To further investigate the model's performance, a number of visualizations were developed including learning curves, activation heatmaps, and actual vs. predicted plots.

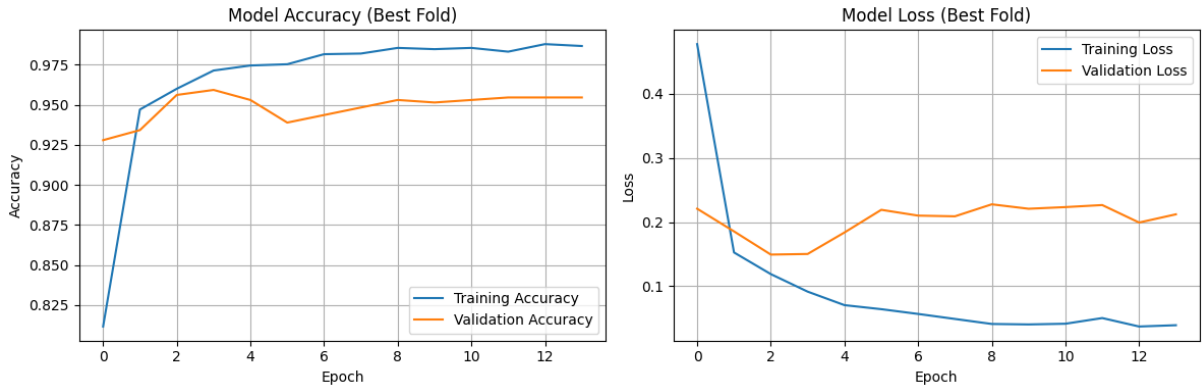

Figure 2: Learning curves showing training and validation accuracy and loss over epochs

Figure 2 Training curves demonstrate the model's learning curve over 100 epochs and early stopping is introduced when the accuracy plateues. The left graph indicates how the validation and training accuracy steadily rise in the early epochs, with the training accuracy reaching almost 0.98 and validation accuracy stabilizing at almost 0.95 with good generalization. The right graph indicates the steep decline in both the training loss and the validation loss, where the training loss continues its decreasing trend and the validation loss plateaus at a low point. Minimal overfitting is evidenced by the slight gap between the training and validation curves. These curves in total indicate that the model is learning well and is achieving high performance on both the training as well as the validation data.

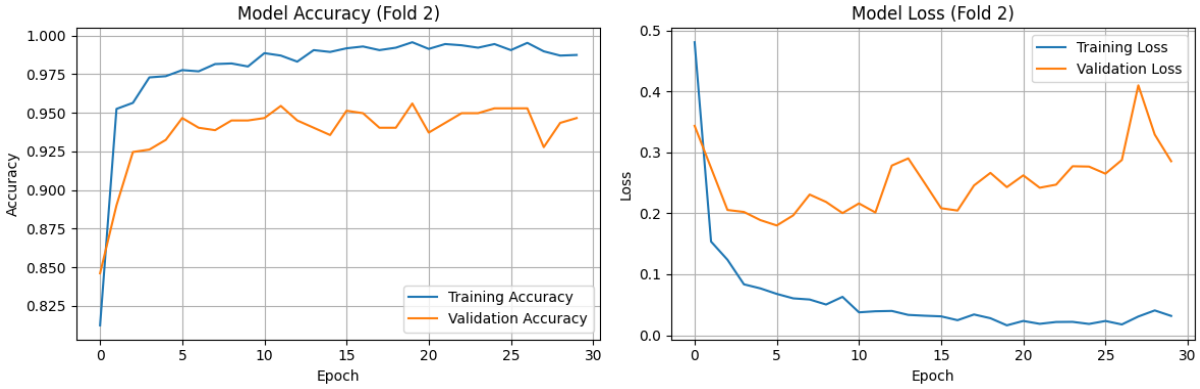

Figure 3: Learning curves for fold 2

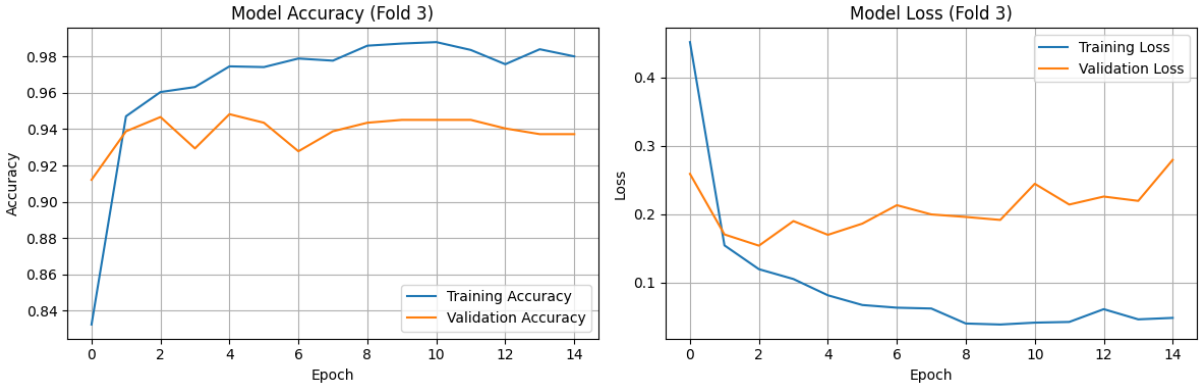

Figure 4: Learning curves for fold 3

Comparatively, the other folds (2, 3, 4, 5) also show strong accuracy on train data but lesser consistency in accuracy on validation as well as specifically on validation loss, where it fluctuates or even rises at times. That is what is expected with cross-validation, as it reflects variation in data splits, but on average all of them ensure that the model is stable and generalizes well no matter what subsets of data it is given.

The best-performing fold has the best learning trend, with rapid rise in training as well as validation accuracy, with these staying at high levels, and with leveling out in validation loss on a low plateau. This indicates good learning, as well as generalizing, happening, with minimal overfitting or instability.

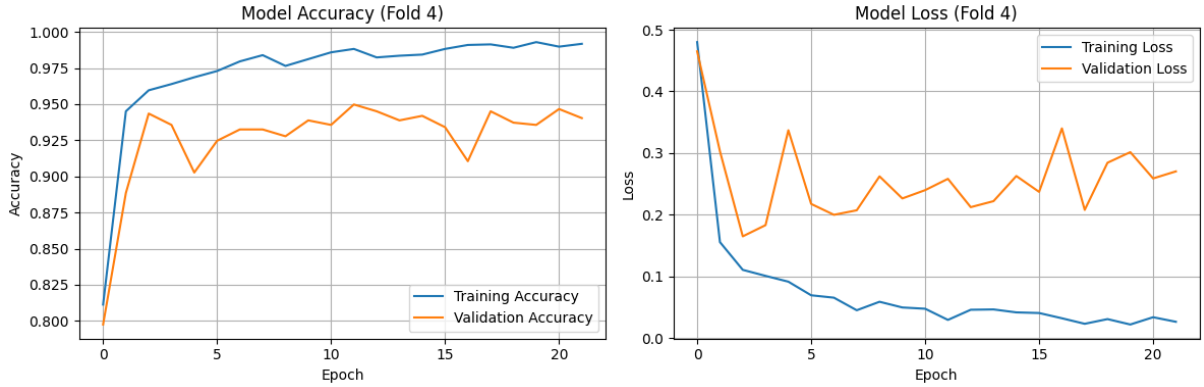

Figure 5: Learning curves for fold 4

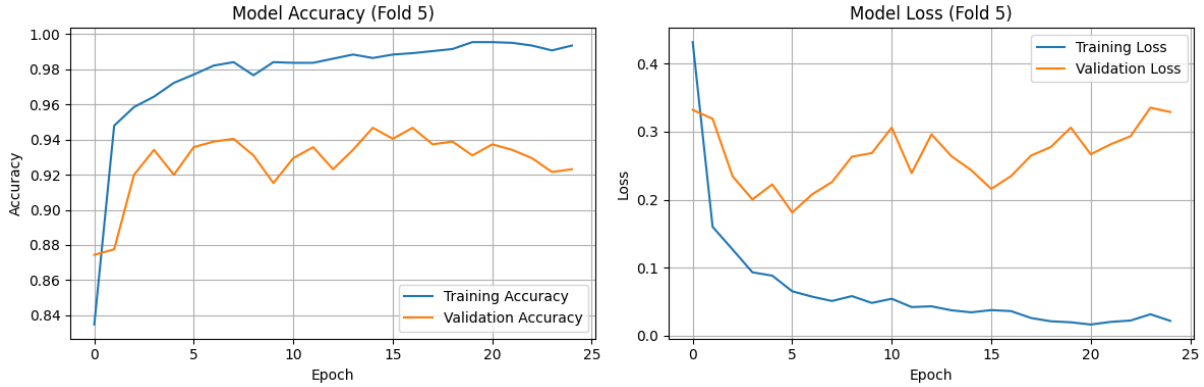

Figure 6: Learning curves for fold 5

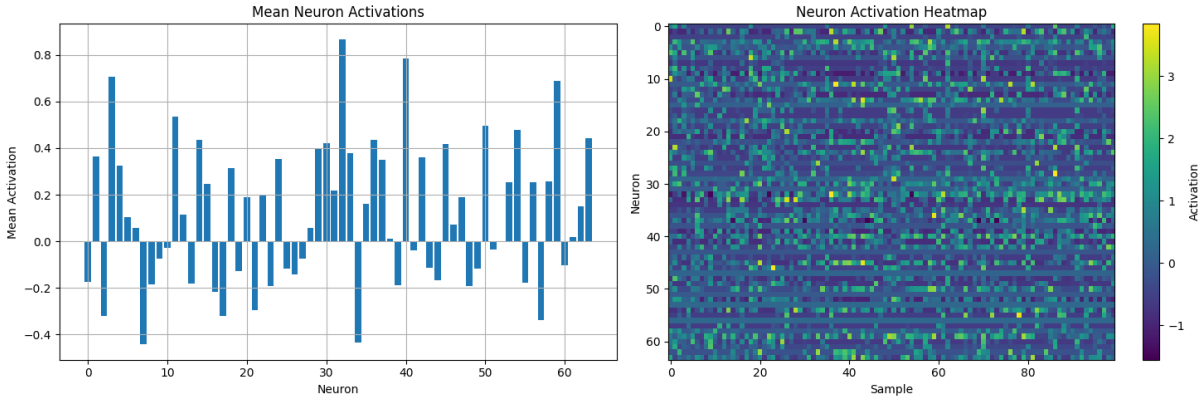

Figure 7: Activation heatmap showing neuron activation patterns in the first hidden layer

Figure 7 The Activation heatmap allows us to see how well a hidden layer within a neural network is performing. Mean neural activation is represented in the graph on the left with point values close to zero, a sign that neurons are not saturating or fully inactivated. This signifies healthy learning as well as proper use of network capacity.

Activation graphs over samples, as well as low overall activations with occasional high spikes, are represented in the heatmap on the right. This is a sign of sparse, rich neuron activations, capable of making the model better at learning complicated patterns within data. Overall, these visualizations ensure healthy, successful activity within neurons in a network.

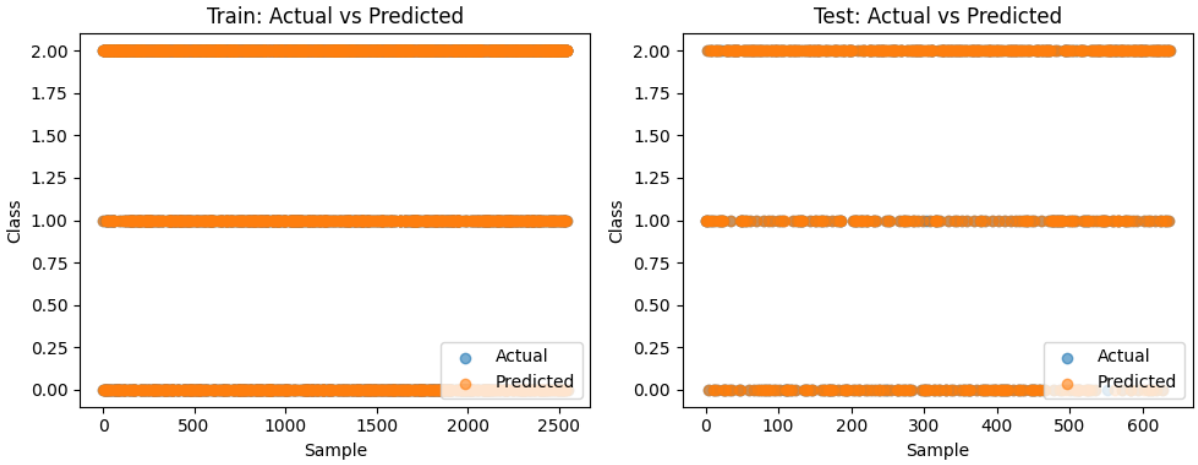

Figure 8: Actual vs. predicted plots for train and test sets

Figure 8 Actual versus predicted plots of the training and test sets indicate the accuracy of the model across three classes. True class (blue) and predicted class (orange) are labeled for each sample, and the nearly complete overlap of the two indicate that the model predictions closely follow the true labels. The good agreement with both the training and the test set indicate generalizability of the model and the fact that the model is not overfitted. The good separation of classes and the absence of glaring misclassifications also indicate the accuracy and robustness of the model in the gene sequence data classification.

Insights: The above charts give general insight into the behavior of the model. The learning curves reflect the steep improvement and the high accuracy rate with little overfitting. The confusion matrix shows the correct proportion of predictions and minimal misclassifications for each of the classes. The neuron activation charts confirm that the neurons in the network are active and well-utilized, allowing effective learning. The actual vs. predicted charts finally reflect the good agreement between true and predicted classes in the training and test data, respectively, vouching for the reliability of the model and good generalizability to new, unseen data. The model performs generally well and steadily.

6 Conclusion

This project was successful in developing an end-to-end machine learning pipeline for the splice-junction gene sequences' classification. Care was undertaken in each stage of the process, from preprocessing to rigorous model selection, hyperparameter tuning, and evaluation. The ultimate ANN model was of satisfactory performance with good accuracy, precision, recall, and specificity for the training and the test dataset. Specifically, the model did not indicate the slightest sign of overfitting nor underfitting, indicating good generalizability to new data. Informative visualizations also aided the model's interpretability and decision-making, such as learning its behavior. Generally speaking, this study highlights the power and reliability of neural networks in biological sequence classification as a good starting point for more research and applications in genomics and related fields at the postgraduate level. The pipeline developed and results presented herein can serve as a model for more biological data analysis with machine learning approaches.

References

[1] B. Alberts, A. Johnson, J. Lewis *et al.*, *Molecular Biology of the Cell.* Garland Science, 2014.

[2] E. T. Wang *et al.*, "Alternative splicing in cancer: implications for diagnosis and therapy," *Oncogene*, vol. 34, no. 1, pp. 1–14, 2015.

[3] M. O. Noordewier, G. G. Towell, and J. W. Shavlik, "Molecular biology (splice-junction gene sequences)," *Machine Learning*, vol. 6, no. 3, pp. 195–225, 1991.

[4] W. McKinney, "pandas: powerful python data analysis toolkit," *Python for Data Analysis*, 2010.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[7] M. O. Noordewier, G. G. Towell, and J. W. Shavlik, "Machine learning applications in genetics and molecular biology," *Machine Learning*, vol. 6, no. 1, pp. 11–32, 1991.

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International conference on machine learning*, pp. 448–456, 2015.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[12] F. Chollet *et al.*, "Keras," *GitHub*, 2015. [Online]. Available: https://github.com/fchollet/keras

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011, section 3.3. Metrics and scoring: quantifying the quality of predictions.

[14] P. S. Foundation, "Python itertools module," 2023. [Online]. Available: https://docs.python.org/3/library/itertools.html

[15] ——, "Python warnings module," 2023. [Online]. Available: https://docs.python.org/3/library/warnings.html

[16] C. R. Harris, K. J. Millman, S. J. van der Walt *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[17] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[18] M. L. Waskom, "Seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.