For questions 1~3:

I selected learning rate and hidden units to carry out the experiments. For learning rate, I set 0.0010, 0.0005, and 0.0001; for hidden units, I set 128, 256, and 512. The following table is the comparison of combinations.

Table 1

	Learning Rate	Hidden Units	Train Acc	Val Acc	Test Acc
0	0.0010	128	77.248677	60.493827	58.064516
1	0.0010	256	79.894180	79.012346	70.967742
2	0.0010	512	73.015873	58.024691	48.387097
3	0.0005	128	75.132275	66.666667	64.516129
4	0.0005	256	77.248677	75.308642	74.193548
5	0.0005	512	66.137566	62.962963	70.967742
6	0.0001	128	72.486772	70.370370	67.741935
7	0.0001	256	73.015873	72.839506	61.290323
8	0.0001	512	78.835979	72.839506	61.290323

Assume Human level can get over 99% accuracy, here is the difference between the values of Human level, Train error, Val error, and Test error and the problems of each row:

Table 2

	Human level	Training err	Val err	Test err	Problem
0	21.75132	16.75485		2.429311	High variance
1	19.10582	0.881834		8.044604	Overfitting to val set
2	25.98413	14.99118		9.637594	High variance Overfitting to val set
3	23.86773	8.465608		2.150538	High variance
4	21.75132	1.940035		1.115094	Generalized best
5	32.86243	3.174603		-8.00478	Test distribution differs
6	26.51323	2.116402		2.628435	Generalized better
7	25.98413	0.176367		11.54918	Overfitting to val set
8	20.16402	5.996473		11.54918	Overfitting to val set

For every row of combinations, we can see that all the errors between human level and training set error are very huge, thus we should consider to improve avoidable bias as the first priority.

Aside avoidable bias, here's the summary of my findings. For the best Model, I would choose row 4 with 0.0005 learning rate and 256 hidden units since the train/val/test accuracy are very close, meaning excellent generalization. Although row 6 also performs with close accuracies/errors, row 4 still performs better than row 6 with closer accuracies/errors. This configuration should be prioritized for

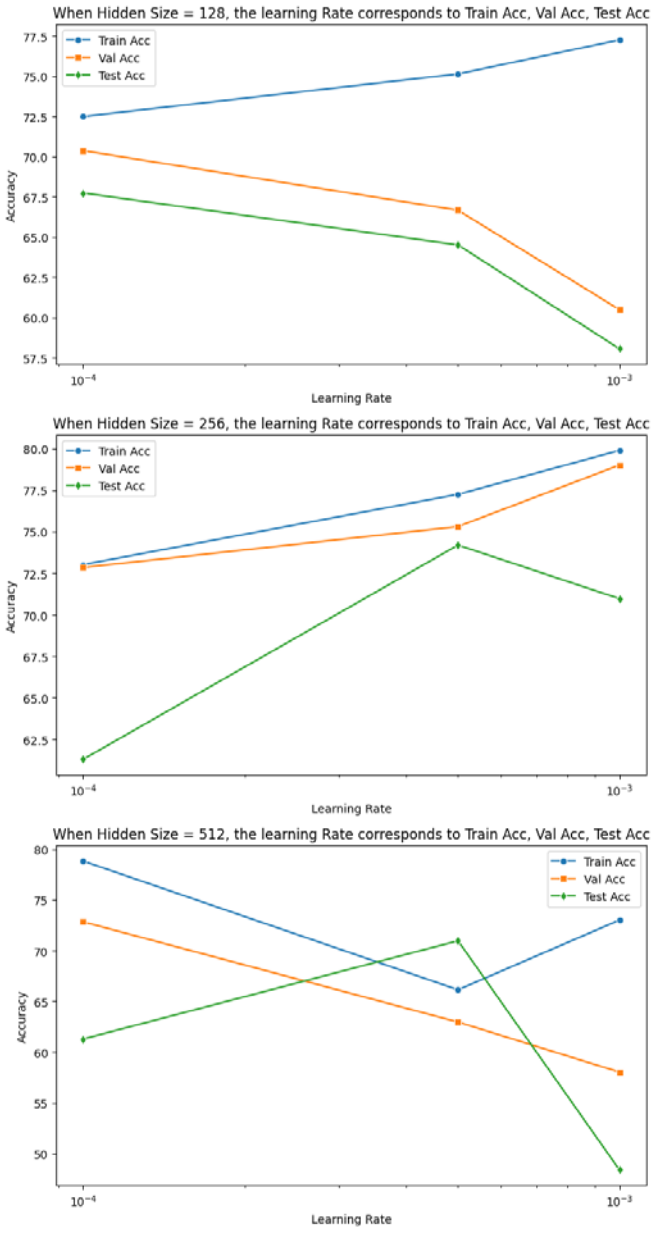
deployment, and work on reducing avoidable bias first.

For the worst overfitting cases, rows 0, 8 and 7 have a large gap between train and validation/test accuracy. Perhaps we need to add dropout, weight decay, data augmentation, or reduce complexity and so on to these rows.

For the underfitting cases, the train accuracy of row 2 and 5 is already low, meaning the model is too weak. In order to fix this, perhaps we can increase hidden units, train longer, or use a different learning rate.

As for high degree of overfitting to validation set, in rows 7 and 8 validation accuracy is fine, but Test accuracy drops significantly. We might need to use more diverse data in validation set.

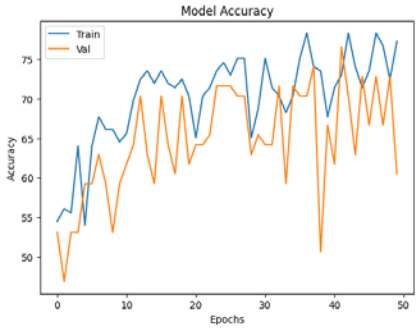
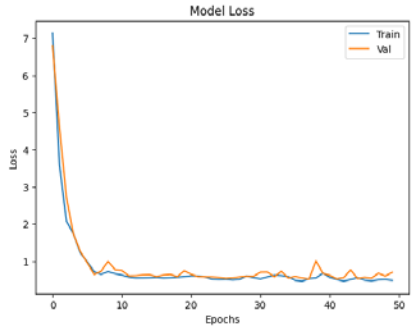
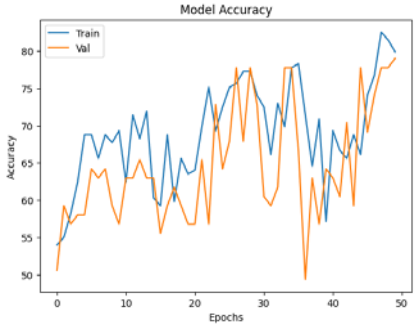
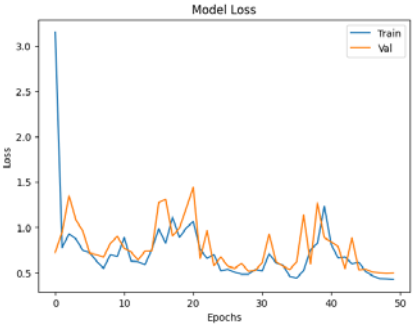
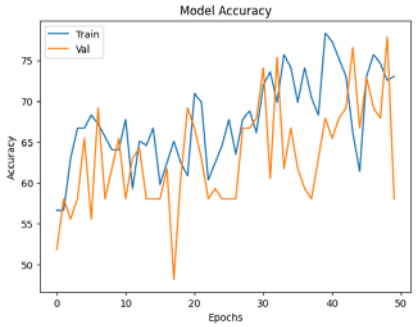
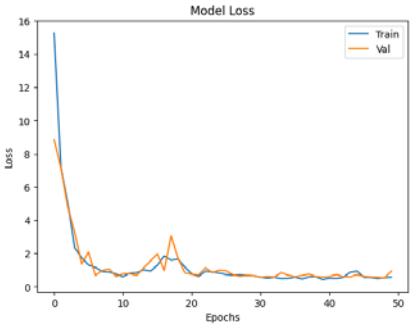
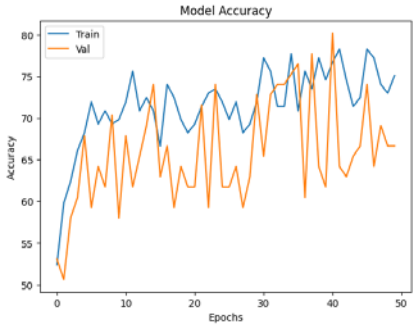
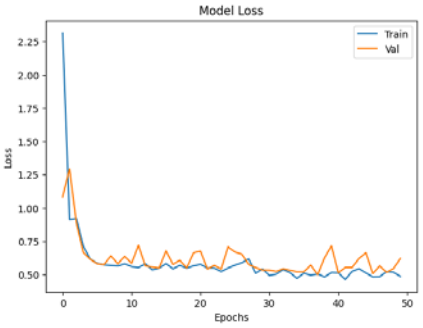
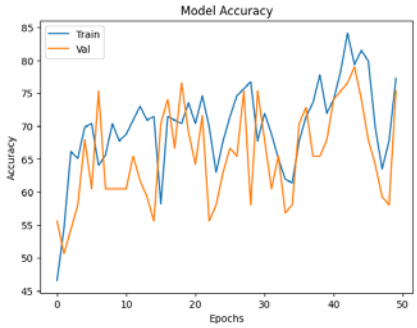
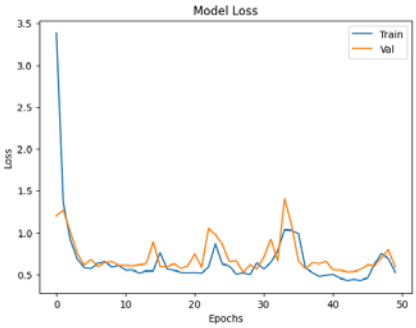
This is the visualization of the learning rate and hidden units metrics:

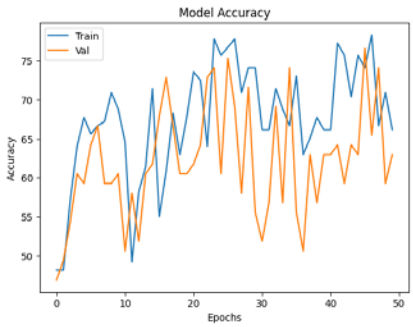
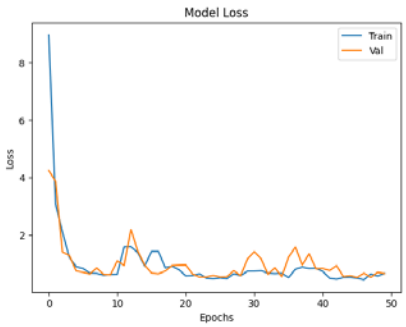
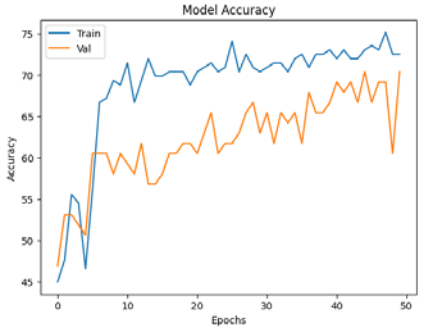
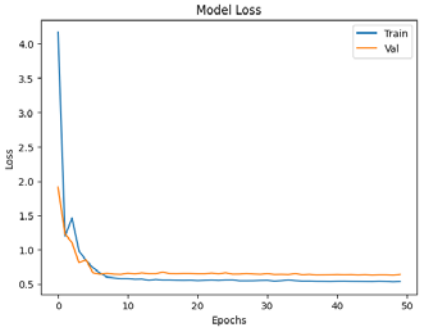
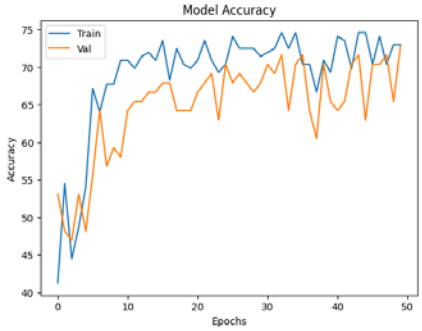
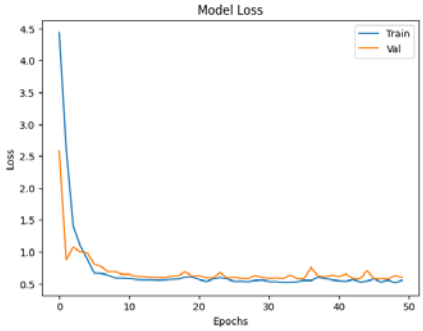
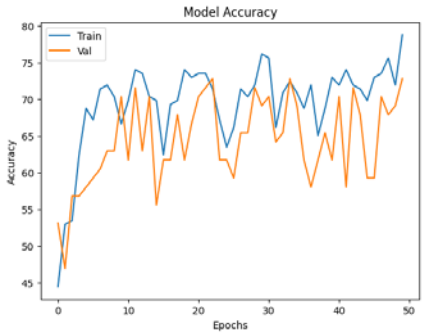
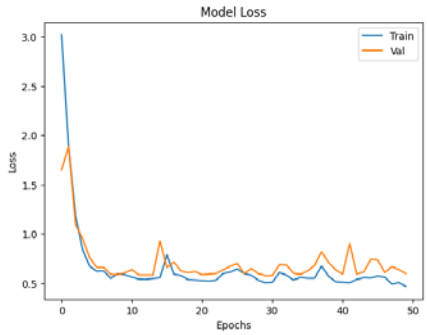


Picture 1

The following table is the training behavior of different models based on Model Accuracy and Model Loss plots.

Table 3

Learning Rate: 0.001, Hidden Size: 128		
Learning Rate: 0.001, Hidden Size: 256		
Learning Rate: 0.001, Hidden Size: 512		
Learning Rate: 0.0005, Hidden Size: 128		
Learning Rate: 0.0005, Hidden Size: 256		

Learning Rate: 0.0005, Hidden Size: 512	 
Learning Rate: 0.0001, Hidden Size: 128	 
Learning Rate: 0.0001, Hidden Size: 256	 
Learning Rate: 0.0001, Hidden Size: 512	 

In general, for model accuracy, the train accuracy is generally higher than the validation accuracy across different settings. Some models show high fluctuations in validation accuracy, indicating high variance during training. For model loss, loss curves generally decrease over epochs, meaning the models are learning.

For question 4:

Methodologies for selecting relevant features in tabular data include filter methods (e.g., correlation, chi-squared), wrapper methods (e.g., recursive feature elimination), and embedded methods (e.g., L1 regularization in linear models, feature importance in tree-based models). Feature selection is crucial for reducing dimensionality, improving model interpretability, decreasing training time, and preventing overfitting, ultimately leading to better generalization and performance. For instance, removing highly correlated features can prevent multicollinearity issues in linear models.

Reference: Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*

For question 5:

TabNet is a deep learning model specifically designed for tabular data. Unlike ANNs, which can struggle with the unstructured nature of tabular data, TabNet utilizes a sequential attention mechanism to select relevant features at each decision step. This allows the model to learn which features are important for different instances. Furthermore, it incorporates feature masking to ensure that only the selected features contribute to the decision-making process in each step. This design choice enhances interpretability by providing feature importance scores and improves performance by focusing on the most informative features, often outperforming traditional ANNs on tabular tasks.

Reference: Arik, S. Ö., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*