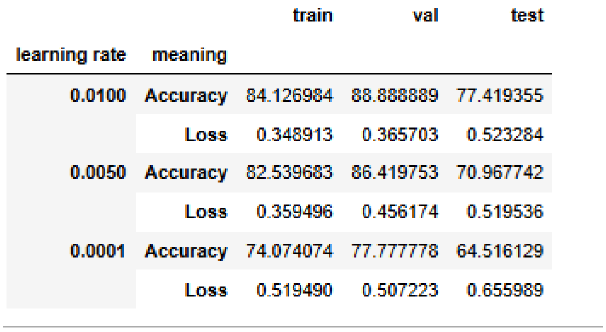
1. Select 2 hyper-parameters of the artificial neural network used in Lab 2 and set 3 different values for each. Perform experiments to compare the effects of varying these hyper-parameters on the loss and accuracy metrics across the training, validation, and test datasets. Present your findings with appropriate tables.

set **learning rate** with value 0.01, 0.005, 0.0001

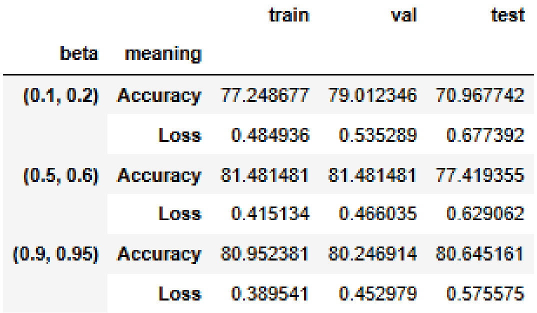
In this task, larger learning rate tends to perform better on train, dev and test dataset. And the smallest gap of performance between test and train set occurs when the learning rate is 0.01, which addresses the model learn feature effectively enough to handle the variety in test data. On the other hand, when learning rate is 0.005 or 0.0001 there are over 10% drop in accuracy, which indicates the learning rates are too slow causing the underfitting.



		train	val	test
learning rate	meaning			
0.0100	Accuracy	84.126984	88.888889	77.419355
	Loss	0.348913	0.365703	0.523284
0.0050	Accuracy	82.539683	86.419753	70.967742
	Loss	0.359496	0.456174	0.519536
0.0001	Accuracy	74.074074	77.777778	64.516129
	Loss	0.519490	0.507223	0.655989

set **Beta** with value (0.1, 0.2), (0.5, 0.6), (0.9, 0.95)

When Beta set to (0.1, 0.2), the model performs well on train and dev set but not test set, which shows the poor generalization. As the Beta increase to (0.5, 0.6) the model seems learn well and perform better than the previous one. Beta equals to (0.9, 0.95) provide the lowest losses overall experiment and there is merely a small gap of performance between train and test indicating the good learning of the model with good generalization.

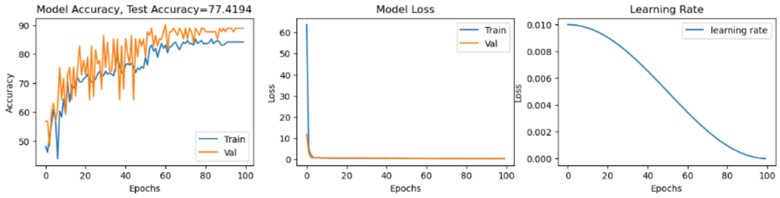


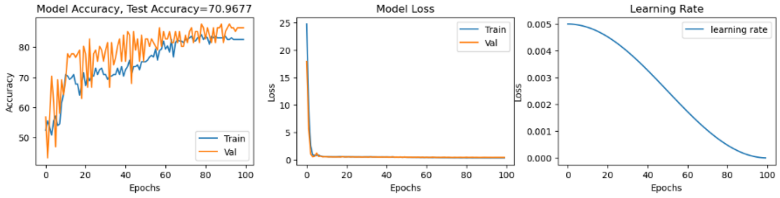
		train	val	test
beta	meaning			
(0.1, 0.2)	Accuracy	77.248677	79.012346	70.967742
	Loss	0.484936	0.535289	0.677392
(0.5, 0.6)	Accuracy	81.481481	81.481481	77.419355
	Loss	0.415134	0.466035	0.629062
(0.9, 0.95)	Accuracy	80.952381	80.246914	80.645161
	Loss	0.389541	0.452979	0.575575

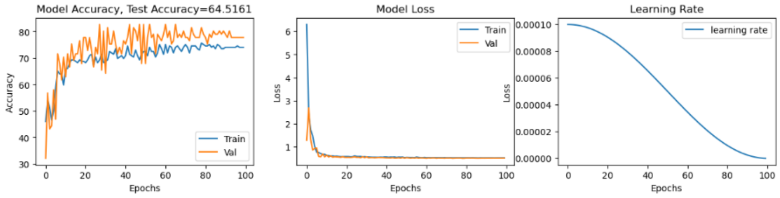
2. Based on your experiments in Question 1, analyze the outcomes. What differences do you observe with the changes in hyper-parameters? Discuss whether these adjustments contributed to improvements in model performance, you can use plots to support your points. (Approximately 100 words.)

   Change Learning Rate

   The larger learning rate seems to reach the lowest loss the fastest and the accuracy still maintain increasing until close to 80%, which means the larger learning rate able the model to jump out the local minimum to find the relatively approximate global minimum. The rest of settings of learning rate show that the accuracies quickly reach the limit and stop growing in the remaining training epochs.

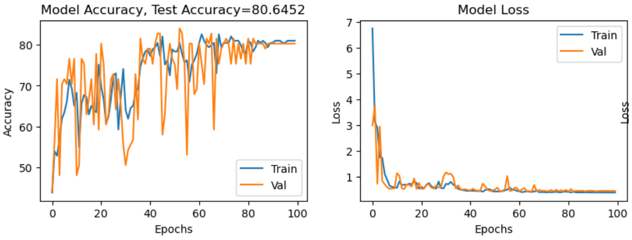


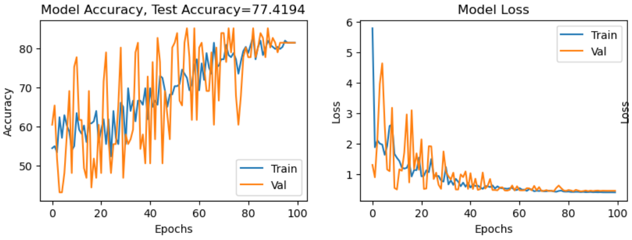


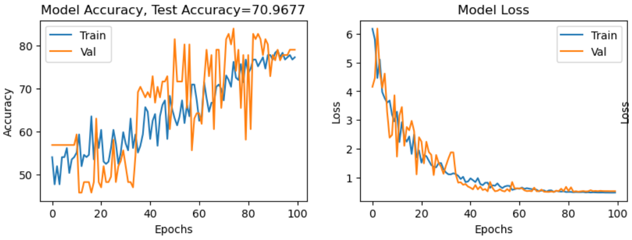


   Change Beta

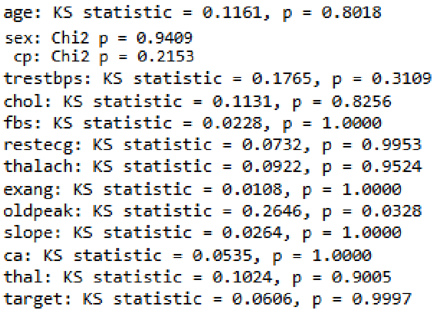
   The image below presents experiments using decreasing values of β. As is well known, larger β values introduce more smoothing, making the model less sensitive to noise, while smaller β values result in less smoothing and greater sensitivity. Consequently, we observe increased noise and fluctuations in the lower plots. Although the loss continues to decrease across all settings, the plots with smaller β values exhibit more pronounced zig-zag patterns compared to the smoother curves seen with higher β values.







3. In Lab 2, you may have noticed a discrepancy in accuracy between the training and test datasets. What do you think causes this occurrence? Discuss potential reasons for the gap in accuracy.



```
age: KS statistic = 0.1161, p = 0.8018
sex: Chi2 p = 0.9409
 cp: Chi2 p = 0.2153
trestbps: KS statistic = 0.1765, p = 0.3109
chol: KS statistic = 0.1131, p = 0.8256
fbs: KS statistic = 0.0228, p = 1.0000
restecg: KS statistic = 0.0732, p = 0.9953
thalach: KS statistic = 0.0922, p = 0.9524
exang: KS statistic = 0.0108, p = 1.0000
oldpeak: KS statistic = 0.2646, p = 0.0328
slope: KS statistic = 0.0264, p = 1.0000
ca: KS statistic = 0.0535, p = 1.0000
thal: KS statistic = 0.1024, p = 0.9005
target: KS statistic = 0.0606, p = 0.9997
```

We first utilized Kolmogorov–Smirnov (KS) Test for continuous features and Chi-square Test for Categorical features between train and test dataset. The result shown below states that there is only one feature “oldpeak” with p-value less than 0.05, which means the other features shows consistency, in turn, we can conclude that the train and test dataset almost come from same distribution. Hence, we can assume reasonably that the key reason causes this phenomenon is the small size of train and test data. The only 189 training data is not enough for deep learning model to learn and only 31 test data is also too small for evaluate the model in general.

4. Discuss methodologies for selecting relevant features in a tabular dataset for machine learning models. Highlight the importance of feature selection and how it can impact model performance. You are encouraged to consult external resources to support your arguments. Please cite any sources you refer to. (Approximately 100 words, excluding reference.)

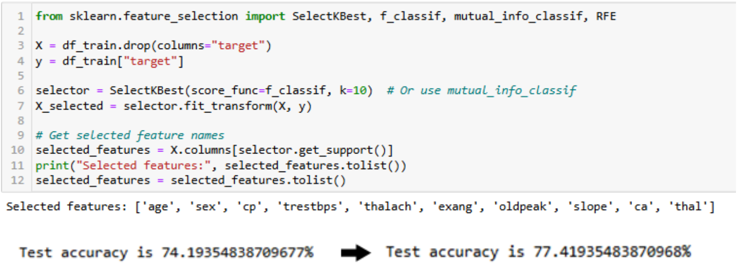
   There are 3 main types of feature selection: filter methods: evaluate the relevance of features based on statistical measures, independent of any machine learning algorithms; wrapper methods: assess subsets of features by actually training models and evaluating their performance, embedded methods: conduct feature selection during the model training process, with algorithms inherently selecting features while constructing the model.

   Due to Hughes Phenomenon [1], also known as the peaking phenomenon, refers to the observation that, for a fixed number of training samples, the performance of a classifier initially improves as the number of features increases, reaching an optimal point. Beyond this point, adding more features leads to a decline in classifier performance, that is to say, appropriate feature selection is essentail.

1. Hughes, Gordon. 1968. 'On the mean accuracy of statistical pattern recognizers', *IEEE transactions on information theory*, 14: 55-63.

(Appendix)

I implement filter method by utilizing univariate selection method to select the feature being more correlated to target. The result shows the 3% increase on test data compared with original one.



```
from sklearn.feature_selection import SelectKBest, f_classif, mutual_info_classif, RFE

X = df_train.drop(columns="target")
y = df_train["target"]

selector = SelectKBest(score_func=f_classif, k=10)  # Or use mutual_info_classif
X_selected = selector.fit_transform(X, y)

# Get selected feature names
selected_features = X.columns[selector.get_support()]
print("Selected features:", selected_features.tolist())
selected_features = selected_features.tolist()
```

```
Selected features: ['age', 'sex', 'cp', 'trestbps', 'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal']
```

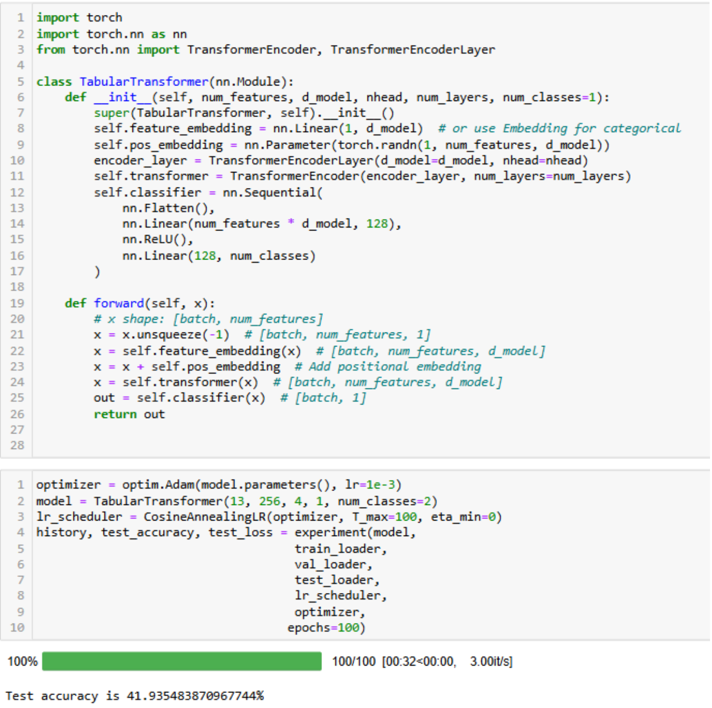
```
Test accuracy is 74.19354838709677%  →  Test accuracy is 77.41935483870968%
```

5. While artificial neural networks (ANNs) are versatile, they may not always be the most efficient choice for handling tabular data. Identify and describe an alternative deep learning model that is better suited for tabular datasets. Explain the rationale behind its design specifically for tabular data, including its key features and advantages. Ensure you to reference any external sources you consult.

   There is a model called TabNet[1] specifically designed for capturing complex relationships between features in tabular data. This model adopts embedding method to conduct feature selection during training, which in turn replaces the traditional method that operate feature engineering before training. Since the embedding method is implemented with attentive transformer, this model provides high interpretability about describing what feature is more and less important. Eventually, the feature transformer is designed to learn the selected features. In addition to the structure of TabNet, the author utilizes self-supervise learning for dealing with missing value in tabular data. The main idea is to adopt self-supervised learning to make model to explore the unknown data automatically for robustness.

1. Arik, S. Ö., & Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(8), 6679-6687. https://doi.org/10.1609/aaai.v35i8.16826

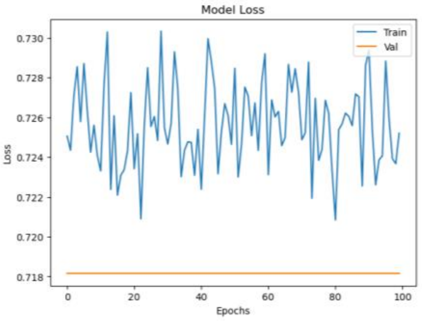
Noticed that such "Deep" network could not handle this tasks well, since I've tried using tabular transformer to solve the problem but the data seems to be too small for capturing the features.

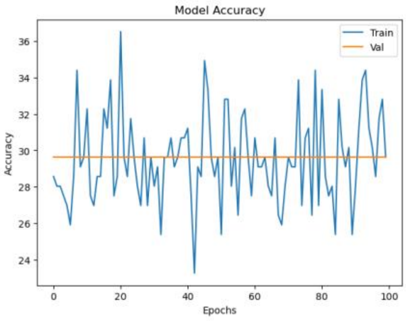


```python
import torch
import torch.nn as nn
from torch.nn import TransformerEncoder, TransformerEncoderLayer

class TabularTransformer(nn.Module):
    def __init__(self, num_features, d_model, nhead, num_layers, num_classes=1):
        super(TabularTransformer, self).__init__()
        self.feature_embedding = nn.Linear(1, d_model)  # or use Embedding for categorical
        self.pos_embedding = nn.Parameter(torch.randn(1, num_features, d_model))
        encoder_layer = TransformerEncoderLayer(d_model=d_model, nhead=nhead)
        self.transformer = TransformerEncoder(encoder_layer, num_layers=num_layers)
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(num_features * d_model, 128),
            nn.ReLU(),
            nn.Linear(128, num_classes)
        )

    def forward(self, x):
        # x shape: [batch, num_features]
        x = x.unsqueeze(-1)  # [batch, num_features, 1]
        x = self.feature_embedding(x)  # [batch, num_features, d_model]
        x = x + self.pos_embedding  # Add positional embedding
        x = self.transformer(x)  # [batch, num_features, d_model]
        out = self.classifier(x)  # [batch, 1]
        return out


```

```python
optimizer = optim.Adam(model.parameters(), lr=1e-3)
model = TabularTransformer(13, 256, 4, 1, num_classes=2)
lr_scheduler = CosineAnnealingLR(optimizer, T_max=100, eta_min=0)
history, test_accuracy, test_loss = experiment(model,
                                               train_loader,
                                               val_loader,
                                               test_loader,
                                               lr_scheduler,
                                               optimizer,
                                              epochs=100)
```

```
100% 100/100 [00:32<00:00,   3.00it/s]

Test accuracy is 41.935483870967744%
```


Model Loss
Train
Val
Loss
Epochs


Model Accuracy
Train
Val
Accuracy
Epochs