Deep Learning Course Final Project

Predicting Brain Tumor Treatment Outcomes

The behavior of brain tumors can be unpredictable and difficult to treat due to the diversity of their clinical presentations and intricate biology. In this project I'll build a deep learning model to predict the treatment outcomes of brain tumors based on various factors such as tumor type, grade, location, etc.

Data Summary/Description

The dataset contains 2000 entries of data about the patients and about their tumors. All of the data is categorical except for the age of the patient.

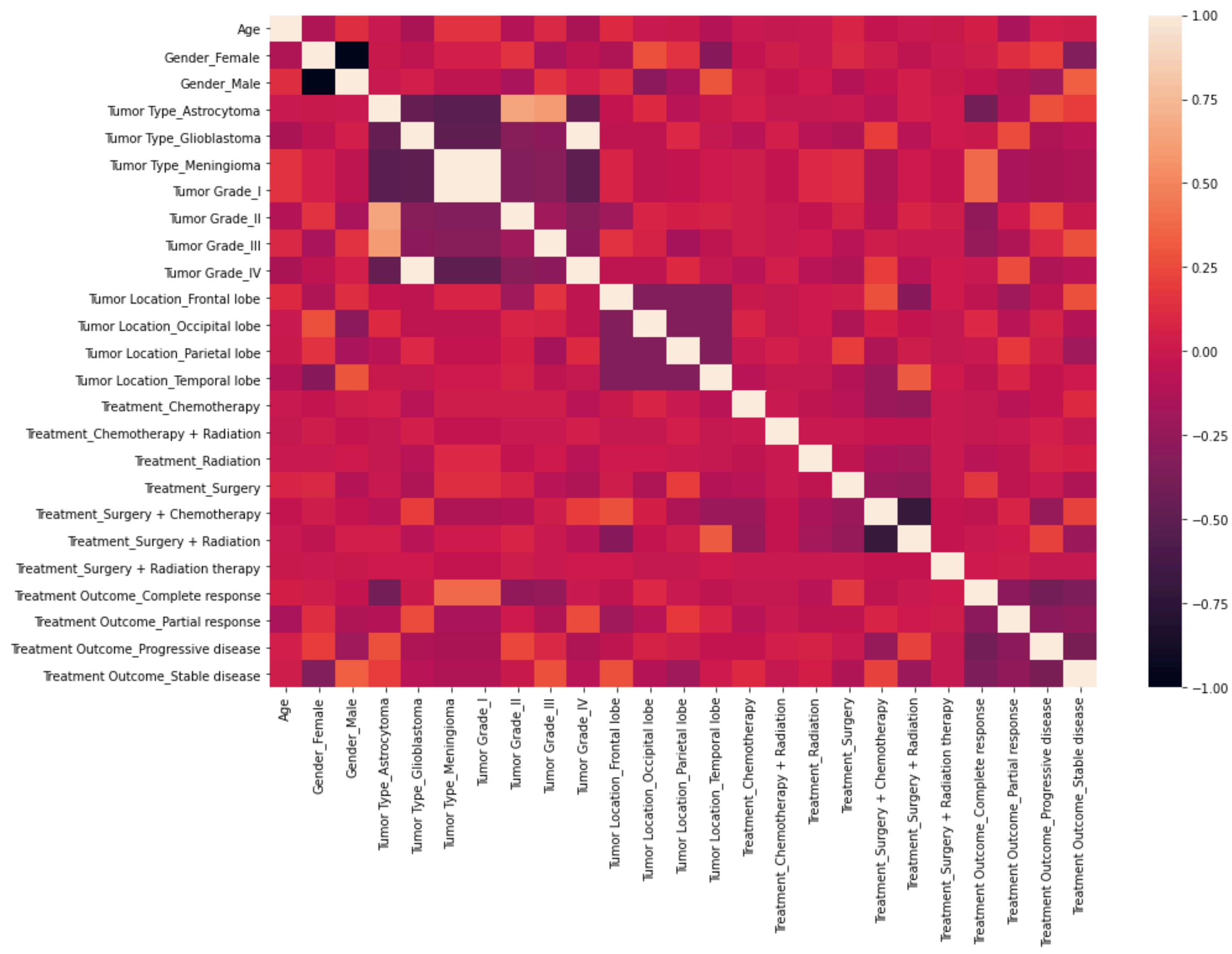
Feature	**Description**
Patient ID	A unique identifier for each patient
Age	Age of patient at time of diagnosis
Gender	Gender of the patient
Tumor Type	Type of tumor (meningioma, astrocytoma, etc)
Tumor Grade	Grade or severity of the tumor (from grade i to grade iv)
Tumor Location	Lobe of the brain where tumor is located
Treatment	Type of treatment recieved
Time to Recurrence	If the tumor recurred, the time to recurrence
Recurrence Site	If the tumor recurred, the recurrence site

Treatment Outcome	The outcome of the treatment

The dependent variable I'm trying to predict would be the treatment outcome, and there are four possible outcomes: complete response, partial response, stable disease, and progressive disease.

Feature Engineering

Because the time to recurrence and recurrence site rows were not applicable to patients who had a complete response for their treatment outcome, those features were dropped. Patient ID was also dropped because it doesn't have any predictive power over the treatment outcome. Age was scaled with a min/max scaler, and all other features were one-hot encoded. The dataset was split into 80/20 test/train datasets.

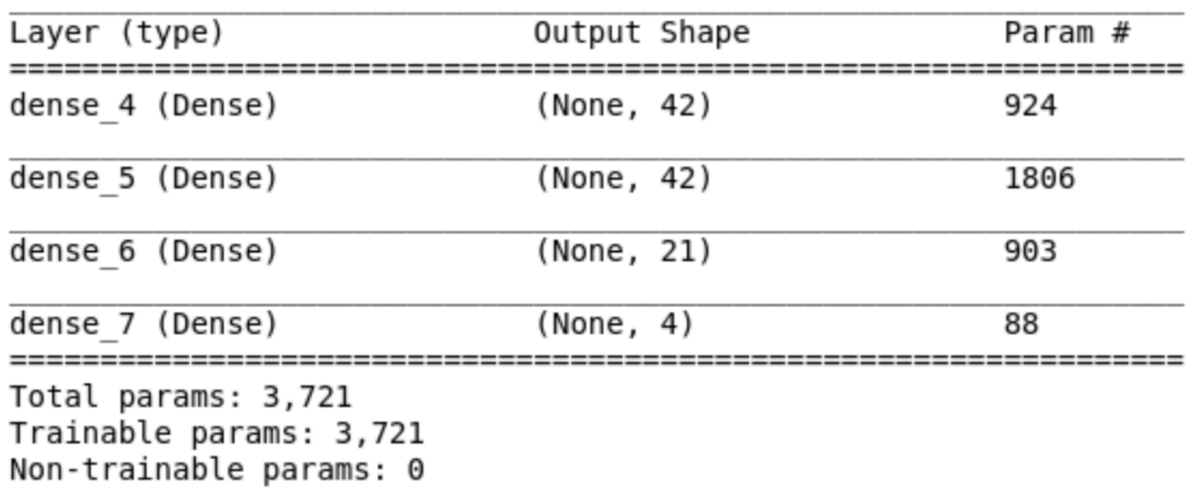


There are some correlations between the features, for example for the treatment outcome of a complete response there's a slight positive correlation to the tumor types being meningioma (which is usually tumor grade i) and a negative correlation with the tumor type being an astrocytoma (which is usually grade ii or iii).

Deep Learning Models

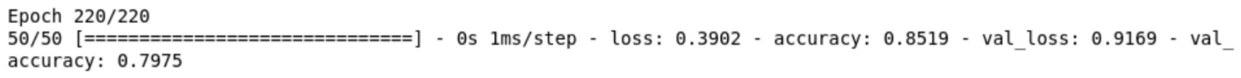
I trained 3 different multilayer perceptron models with different parameters. There are 21 features total for the input layer and 4 features for the output layer (4 different possible outcomes).

Model 1:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_4 (Dense)              (None, 42)                924       
_________________________________________________________________
dense_5 (Dense)              (None, 42)                1806      
_________________________________________________________________
dense_6 (Dense)              (None, 21)                903       
_________________________________________________________________
dense_7 (Dense)              (None, 4)                 88        
=================================================================
Total params: 3,721
Trainable params: 3,721
Non-trainable params: 0
```

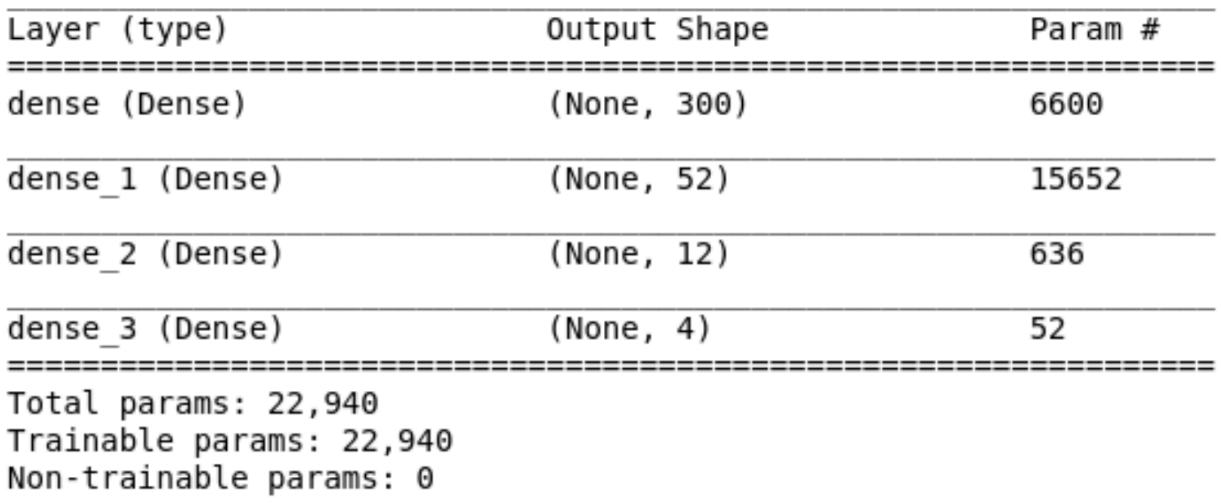
There is an input layer, two hidden dense layers, and then an output layer. I kept all the layers small, with two layers having 42 neurons, one having 21, and the last having 4 for a total of 3,721 parameters to train. All layers had a ReLU activation function except for the last which had a softmax activation. I ran the model for 220 epochs.



```
Epoch 220/220
50/50 [==============================] - 0s 1ms/step - loss: 0.3902 - accuracy: 0.8519 - val_loss: 0.9169 - val_
accuracy: 0.7975
```

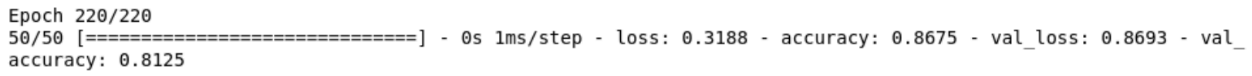
It had a fairly high accuracy on the training set, but a lower accuracy on the test set.

Model 2



```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 300)               6600
_________________________________________________________________
dense_1 (Dense)              (None, 52)                15652
_________________________________________________________________
dense_2 (Dense)              (None, 12)                636
_________________________________________________________________
dense_3 (Dense)              (None, 4)                 52
=================================================================
Total params: 22,940
Trainable params: 22,940
Non-trainable params: 0
```

For this model I kept the overall structure the same, but increased the number of neurons in each layer. There are 300 in the first layer, 52 in the second, 12 in the third, and 4 for the output layer with a total of 22,940 parameters to train. I also ran this for 220 epochs.



```
Epoch 220/220
50/50 [==============================] - 0s 1ms/step - loss: 0.3188 - accuracy: 0.8675 - val_loss: 0.8693 - val_
accuracy: 0.8125
```

This model had a higher accuracy on both the training and test set, though the improvement was higher on the test set.

Model 3

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 300)               6600
_________________________________________________________________
dropout (Dropout)            (None, 300)               0
_________________________________________________________________
dense_9 (Dense)              (None, 52)                15652
_________________________________________________________________
dropout_1 (Dropout)          (None, 52)                0
_________________________________________________________________
dense_10 (Dense)             (None, 12)                636
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 52
=================================================================
Total params: 22,940
Trainable params: 22,940
Non-trainable params: 0
```

This model has the same structure as the second model, but I added two dropout layers (with dropout=0.25).

```
Epoch 220/220
50/50 [==============================] - 0s 2ms/step - loss: 0.4130 - accuracy: 0.8425 - val_loss: 0.7455 - val_
accuracy: 0.8150
```

The accuracy on the test set is slightly higher for this model, and closer to the accuracy for the training set.

Conclusions

The best model was the third model I trained with more neurons and dropout layers to help overfitting to the training set. It may be worth training more models with more layers and/or neurons to see if that improves the accuracy any more. It may also be worth testing different activation functions on the hidden layers to see if that improves accuracy.