U4A3-NN-Titanic

December 1, 2020

```
Author  - Sreejith S
U4A3    - Neural Networks- Titanic
Dataset - https://www.kaggle.com/c/titanic/data
```

11.1 Read about the titanic dataset - [link].

Try to understand the features, the type of features (continuous values, categorical) and the target variable. Download the train.csv and test.csv files. Try to plot each of the features and its influence on the target variable. You can use python or spreadsheet to make the plots. Make your interpretations and try to answer the following questions using the relevant plots.

Feature Engineering

Link

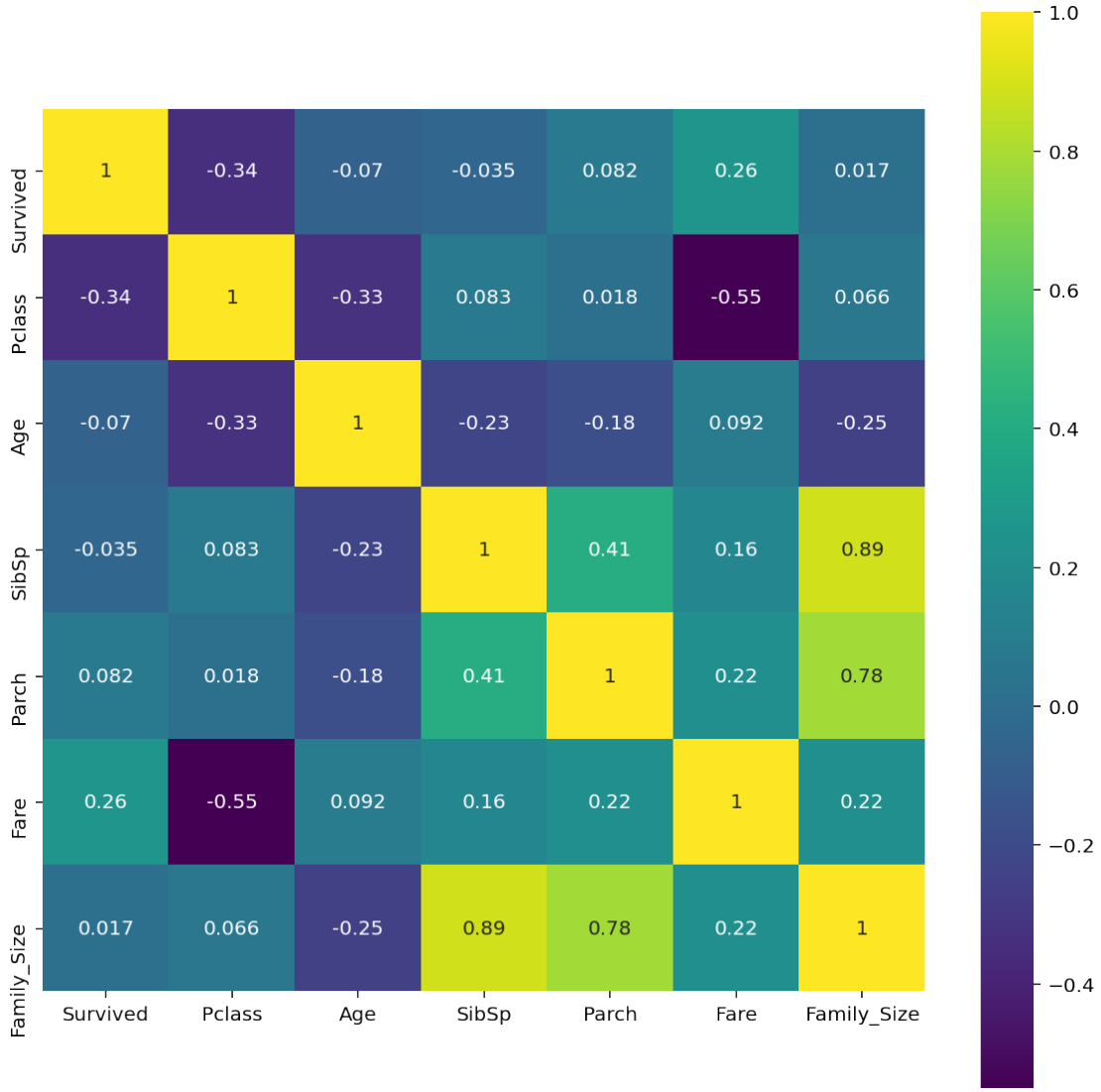
We'll select a few feature engineering techniques from above link as well.

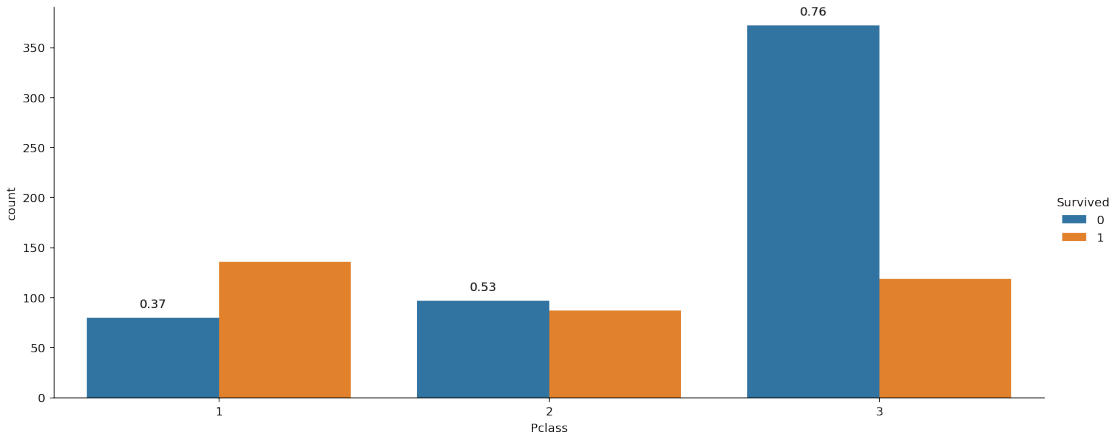
11.1.1 [1 MARK] Which features, in your interpretation, would be good predictors for the survival rate in titanic?

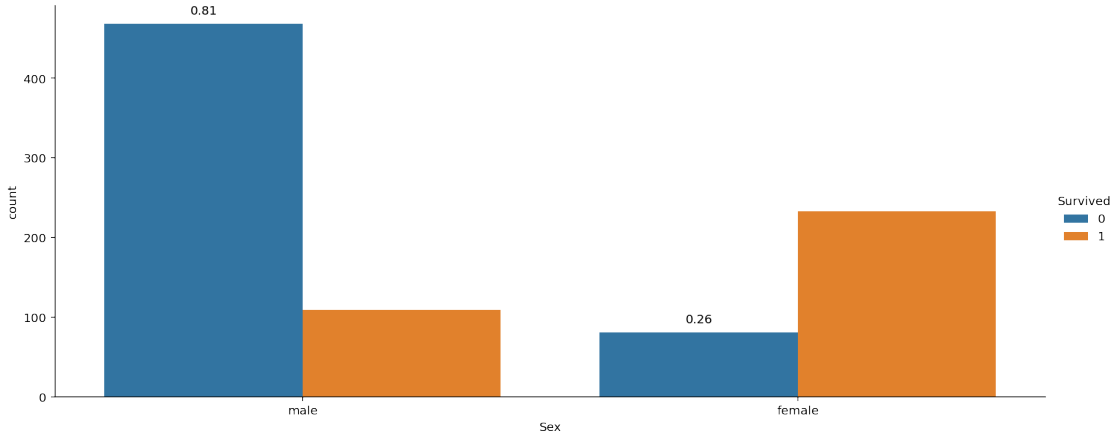
Let's start by looking at the correlation between numerical data. `pandas.DataFrame.corr()` can be used to investigate linear relations between columns, but there might be non-linear relationships too. One thing that is evident from the correlation matrix is that `Pclass` has a strong linear relation with `Survived`.

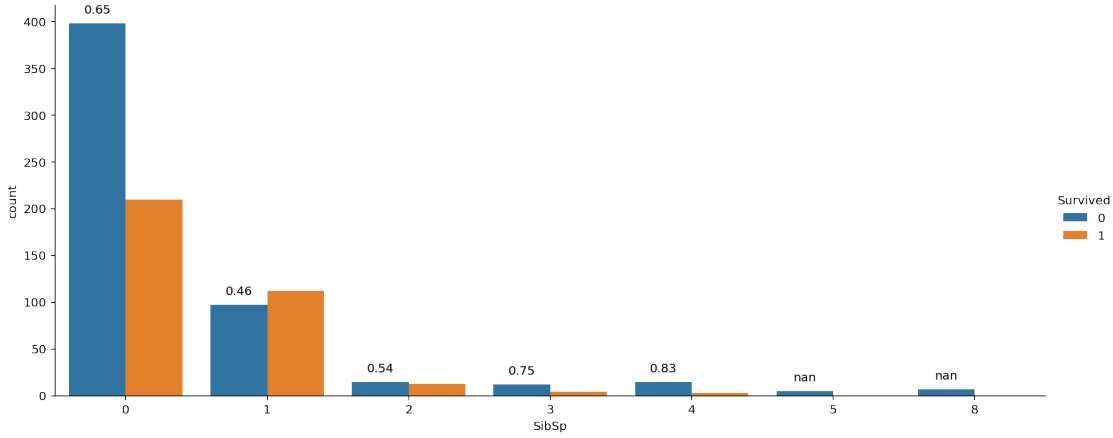
`Pclass`, `Sex`, `SibSp`, `Parch`, `Family_Size`, `Fare`, `Age`, `Embarked`, `Cabin` (Considering Deck) would be good indicators for the survival rate in the Titanic. Passengers travelling in first class, and those paying higher fares had lower mortality rate that others. While mortality rate for men was `80%`, that of women was `26%`. Having no family members onboard or having too many family members reduced the chance of survival.

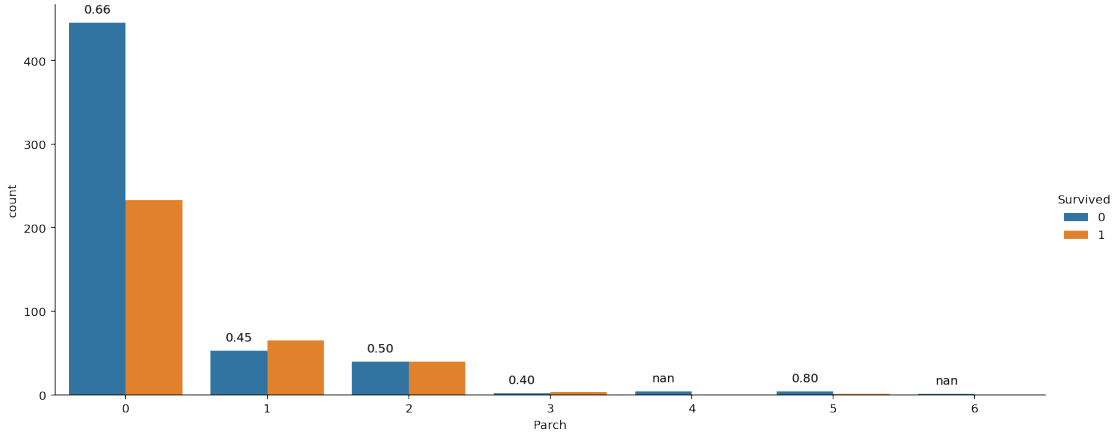
Those with no cabins (third class passengers) had the highest mortality rates. After extracting `Deck` from `Cabin` data, it can be observed that some decks had better survival rates.

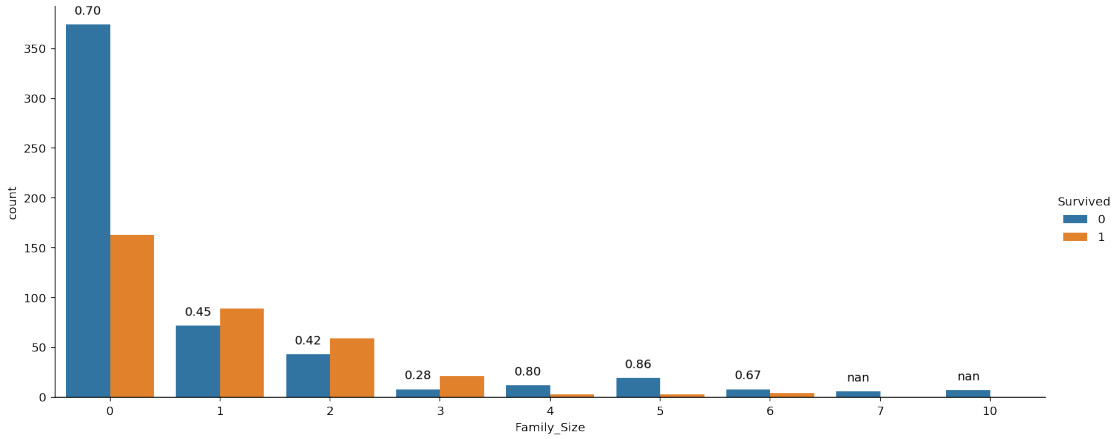

Survived
Pclass
Age
SibSp
Parch
Fare
Family_Size
1 -0.34 -0.07 -0.035 0.082 0.26 0.017
-0.34 1 -0.33 0.083 0.018 -0.55 0.066
-0.07 -0.33 1 -0.23 -0.18 0.092 -0.25
-0.035 0.083 -0.23 1 0.41 0.16 0.89
0.082 0.018 -0.18 0.41 1 0.22 0.78
0.26 -0.55 0.092 0.16 0.22 1 0.22
0.017 0.066 -0.25 0.89 0.78 0.22 1
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

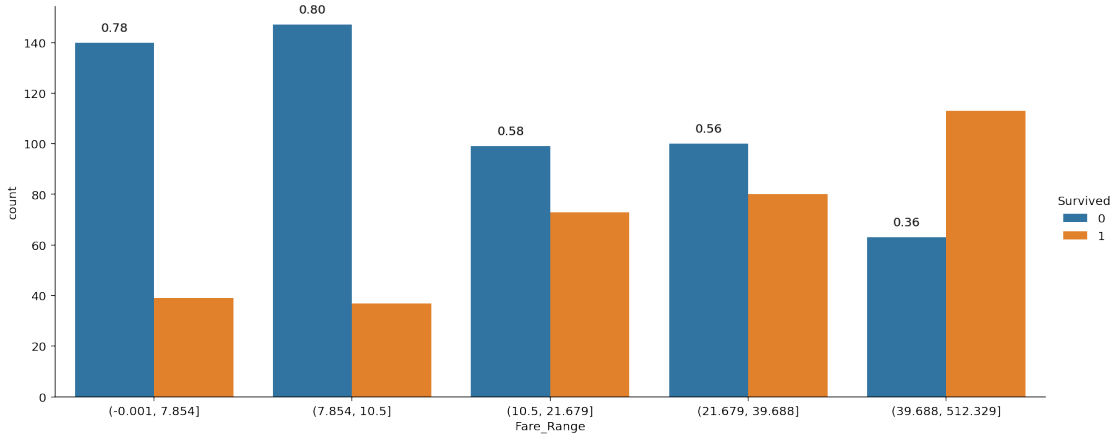

count
Pclass
Survived
0
1
0.37
0.53
0.76
0
50
100
150
200
250
300
350
1
2
3

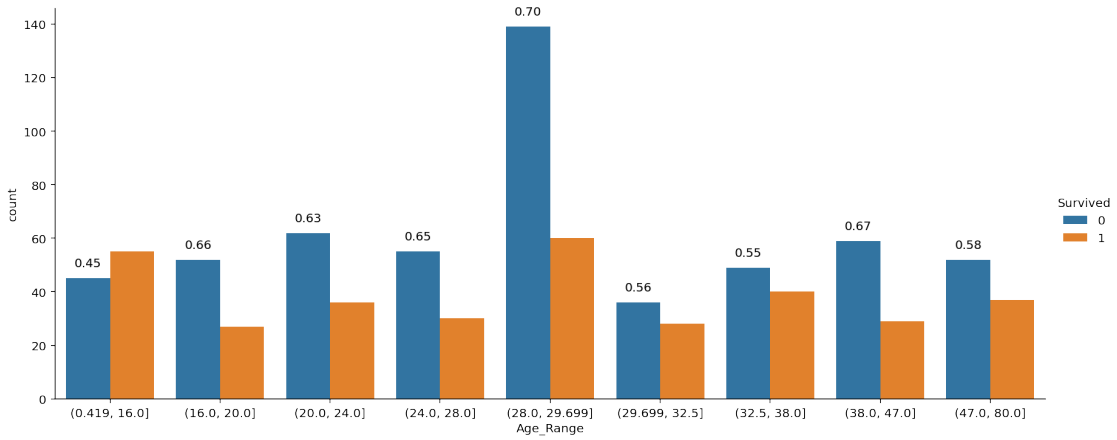

0.81
0.26
count
Sex
male
female
Survived
0
1

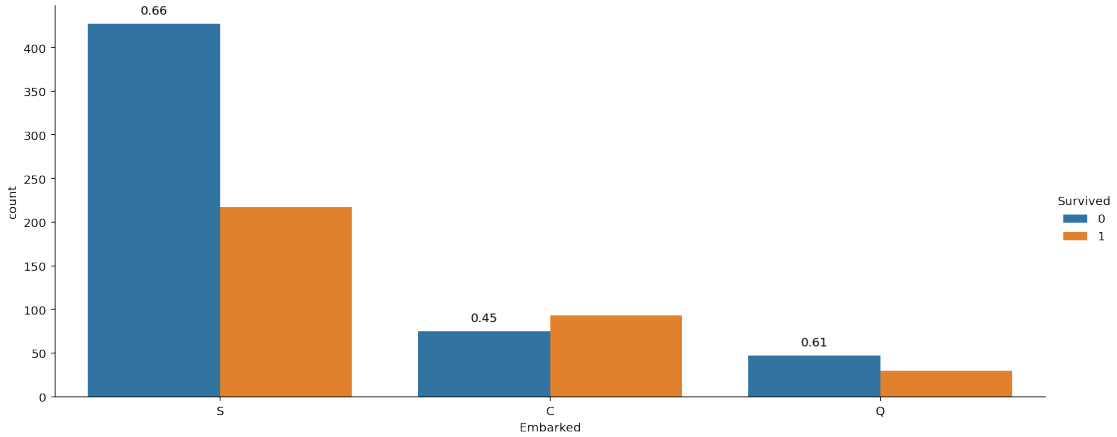

0.65
0.46
0.54
0.75
0.83
nan
nan
count
SibSp
Survived
0
1

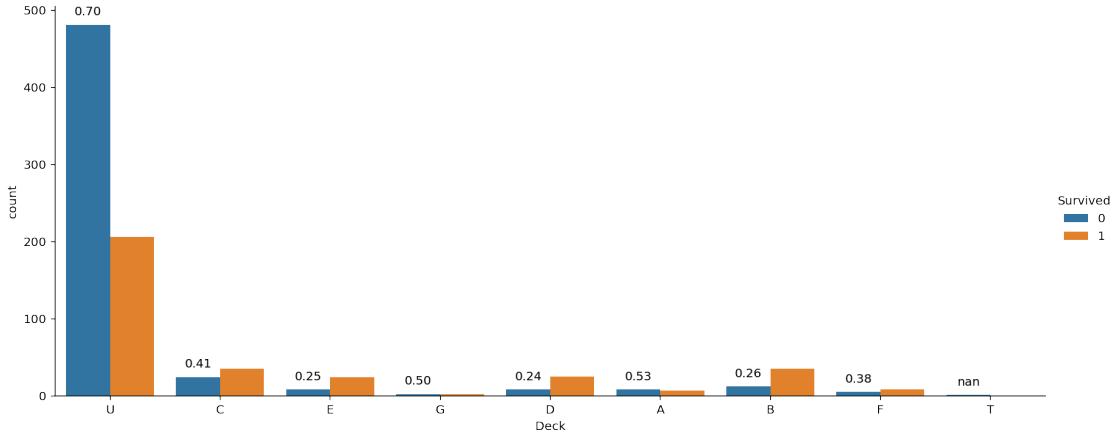

0.66
0.45
0.50
0.40
nan
0.80
nan
count
Parch
Survived
0
1


0.70
0.45
0.42
0.28
0.80
0.86
0.67
nan
nan
count
Family_Size
Survived
0
1


0.78
0.80
0.58
0.56
0.36
count
(-0.001, 7.854]
(7.854, 10.5]
(10.5, 21.679]
(21.679, 39.688]
(39.688, 512.329]
Fare_Range
Survived
0
1


0.45
0.66
0.63
0.65
0.70
0.56
0.55
0.67
0.58
count
(0.419, 16.0]
(16.0, 20.0]
(20.0, 24.0]
(24.0, 28.0]
(28.0, 29.699]
(29.699, 32.5]
(32.5, 38.0]
(38.0, 47.0]
(47.0, 80.0]
Age_Range
Survived
0
1





11.1.2 [1 MARK] Which features, in your interpretation, are not good predictors?

`PassengerId`, `Name` and `Ticket` can be dropped as they wouldn't be good indicators. Even though, surnames can be extracted from `Name` which might be an indicator of of social status and gender. Further exploration has to be done in order to understand whether `Name` and `Ticket` are good predictors. As of now we can ignore them.

11.1.3 What is the size of the train dataset and test dataset?

[129]:
```
print("len(train) = ", len(df_train))
print("len(test) = ", len(df_test))
```

```
len(train) =  891
len(test) =  418
```

11.1.4 [1 MARK] Would you standardize or normalize the input features, or just use the raw features, or try out all different ways and then decide based on final performance evaluation?

The features have different ranges, this means stadardizing or normalizing might help.

11.2 Create a new colab/jupyter notebook and build a multi-layer perceptron (ANN) model

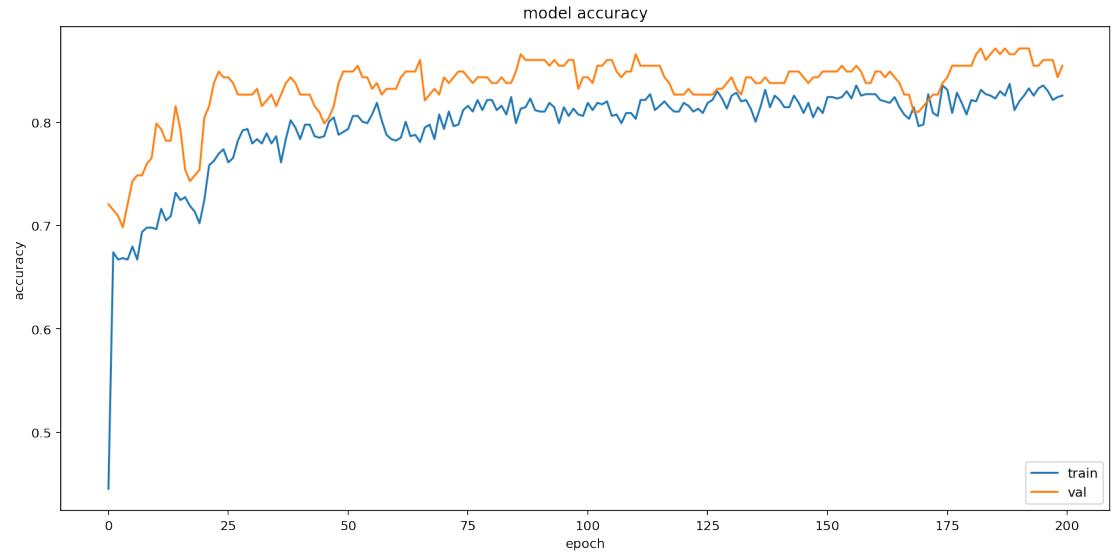
Predict the target variable using (a) all the features (b) only the features that you think are good predictors. You could try to standardize or normalize the features based on your answer to (11.1.4). Report the performance evaluation of these models using standard evaluation metrics for classification problems. Answer the following questions by writing down the answers as well as putting down the screenshots that are relevant to the answer.

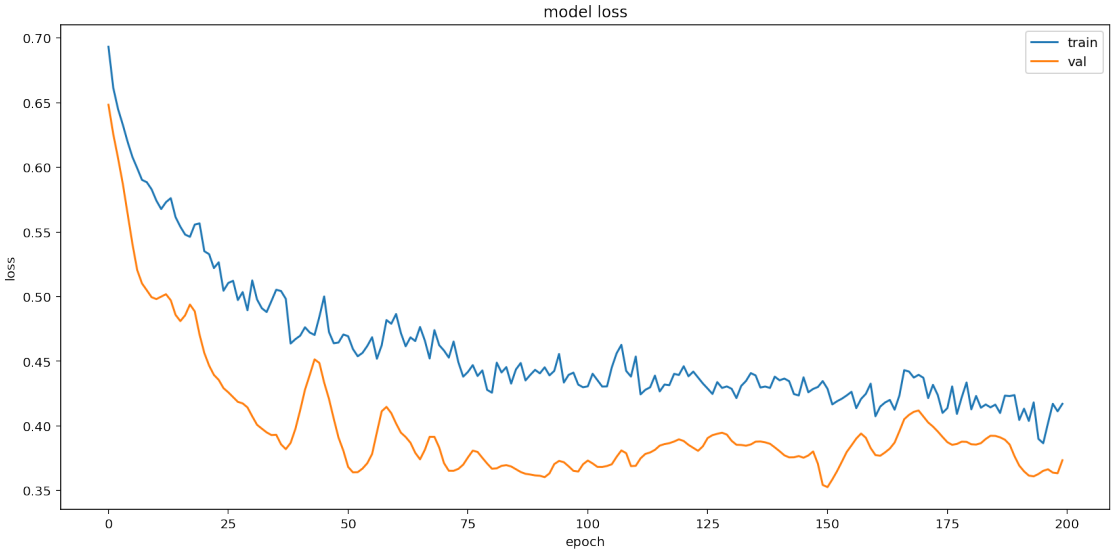
11.2.1 [1 MARK] What is the model loss parameter that you set?

`loss='binary_crossentropy'` is used since this is a binary classification task. It is the average of `categorical_crossenttropy` loss on several two-category tasks.

11.2.2 [1 MARK] What are the different values of epochs and batch sizes that you tried? Is there any effect on the performance of the model when you change these epochs and batch sizes? Write down a relevant table to answer this question.

epoch	batch size	loss	binary_accuracy	val_loss	val_accuracy
500	700	0.2820	0.8876	0.5194	0.8101
200	700	0.3368	0.8652	0.3253	0.8939
100	700	0.3836	0.8357	0.3633	0.8659
50	700	0.4155	0.8244	0.3330	0.8659
200	512	0.2055	0.9171	0.5988	0.8268
200	128	0.2032	0.9143	0.7127	0.7933





11.2.3 [1 MARK] Which feature set is/are found to be best predicting the target variable? What is the best performance that you observe?

`Pclass`, `Sex`, `SibSp`, `Parch`, `Family_Size`, `Fare`, `Age`, `Embarked`, `Deck` are best at predicting target varibles. Best performance obeseved was 0.8939

11.2.4 [1 MARK] What are the different network architectures that you tried out, and what are their performances? Put down this in a table below listing the number of hidden layers, the number of neurons/nodes in each of these layer, and their corresponding performances.

```
20->relu->20->relu->100->relu->50->softmax->2 :
loss: 0.4222 - binary_accuracy: 0.8244 - val_loss: 0.3363 - val_binary_accuracy: 0.8659
```

```
20->relu->20->relu->100->relu->50->softmax->2 :
loss: 0.3790 - binary_accuracy: 0.8469 - val_loss: 0.3454 - val_binary_accuracy: 0.8771

20->relu->20->relu->100->relu->200->relu->100->relu->50->softmax->2 :
loss: 0.3368 - binary_accuracy: 0.8652 - val_loss: 0.3253 - val_binary_accuracy: 0.8939 (epoch=2

20->relu->100->relu->300->relu->512->relu->256->relu->128->softmax->2 :
loss: 0.3152 - binary_accuracy: 0.8792 - val_loss: 0.4219 - val_binary_accuracy: 0.8324 (epoch=2

20->relu->128->relu->128->relu->256->relu->256->relu->512->relu->128->relu->64->softmax->2 :
loss: 0.3000 - binary_accuracy: 0.8778 - val_loss: 0.3893 - val_binary_accuracy: 0.8380 (epoch=2

20->relu->128->relu->128->relu->512->relu->512->relu->1024->relu->256->relu->64->softmax->2 :
loss: 0.3851 - binary_accuracy: 0.8497 - val_loss: 0.3935 - val_binary_accuracy: 0.8827 (epoch=1
```

11.2.5 [1 MARK] Which is the ANN network architecture that performed the best?

`20->relu->20->relu->100->relu->200->relu->100->relu->50->softmax->2` trained for `epochs=200` `with batch_size=700` performed best with `validation accuracy of` 89%

11.3 [2 MARKS] What are the new things that you learned by doing this assignment. List down at least 3 bullet points.

1. Adding dropout layers can reduce overfitting
2. Removing `NaN` values from data set is essential for training. This can be either down by `dropna()` or replacing with mean
3. Higher epoch can lead to overfitting
4. Learned to plot and interpret `seaborn.catplot()`

[ ]: