Lincoln Innes

19 Decmber 2020

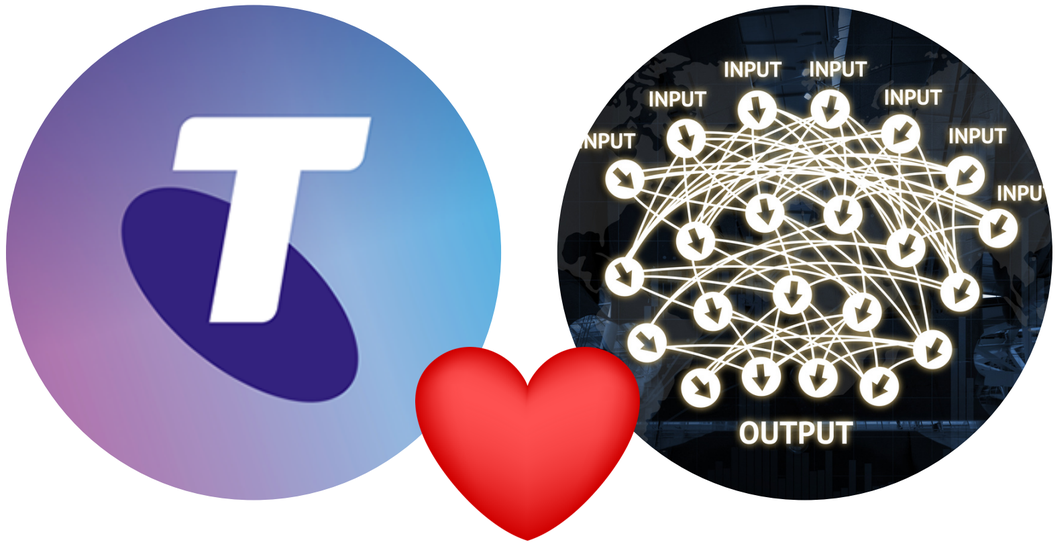
Stop The Churn

Detecting telco customers who are likely to 'churn'

Business Problem

Telecommunications is a saturated market. Growth is by stealing customers.

Customers 'churn' when they cycle through providers.
Need: Identify customers before they 'churn' so they can be retained in a cost effective way.



Telco

Deep Learning

Deep Learning Application: Analyse customer meta-data to determine if they will 'churn'.

Customer Data

What are we working with?

Data from company database
+ Routine report

List of customers to send rention offers

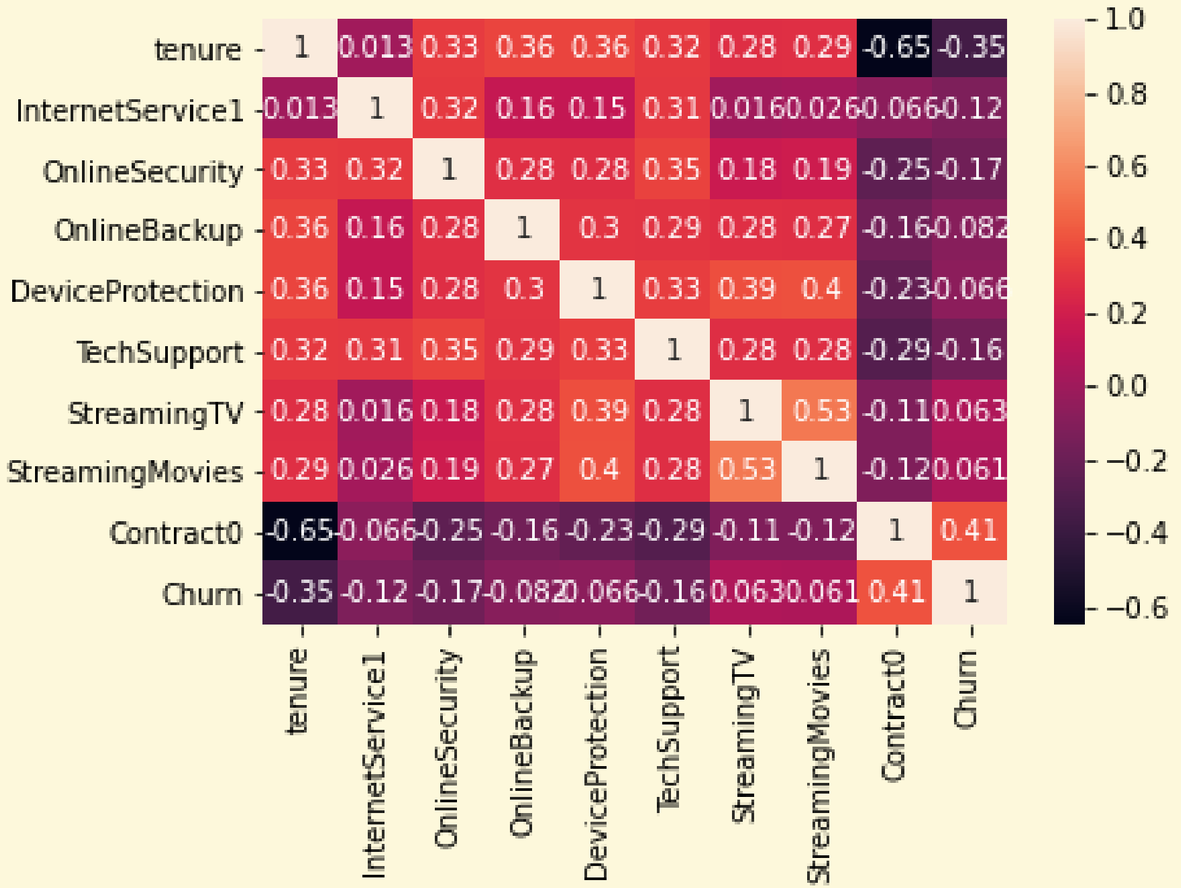
Data for this project from:
https://www.kaggle.com/blastchar/telco-customer-churn

Data Processing

What do we do with the data?

Start with a correlation study.

Identify key data + Eliminate unnecessary data



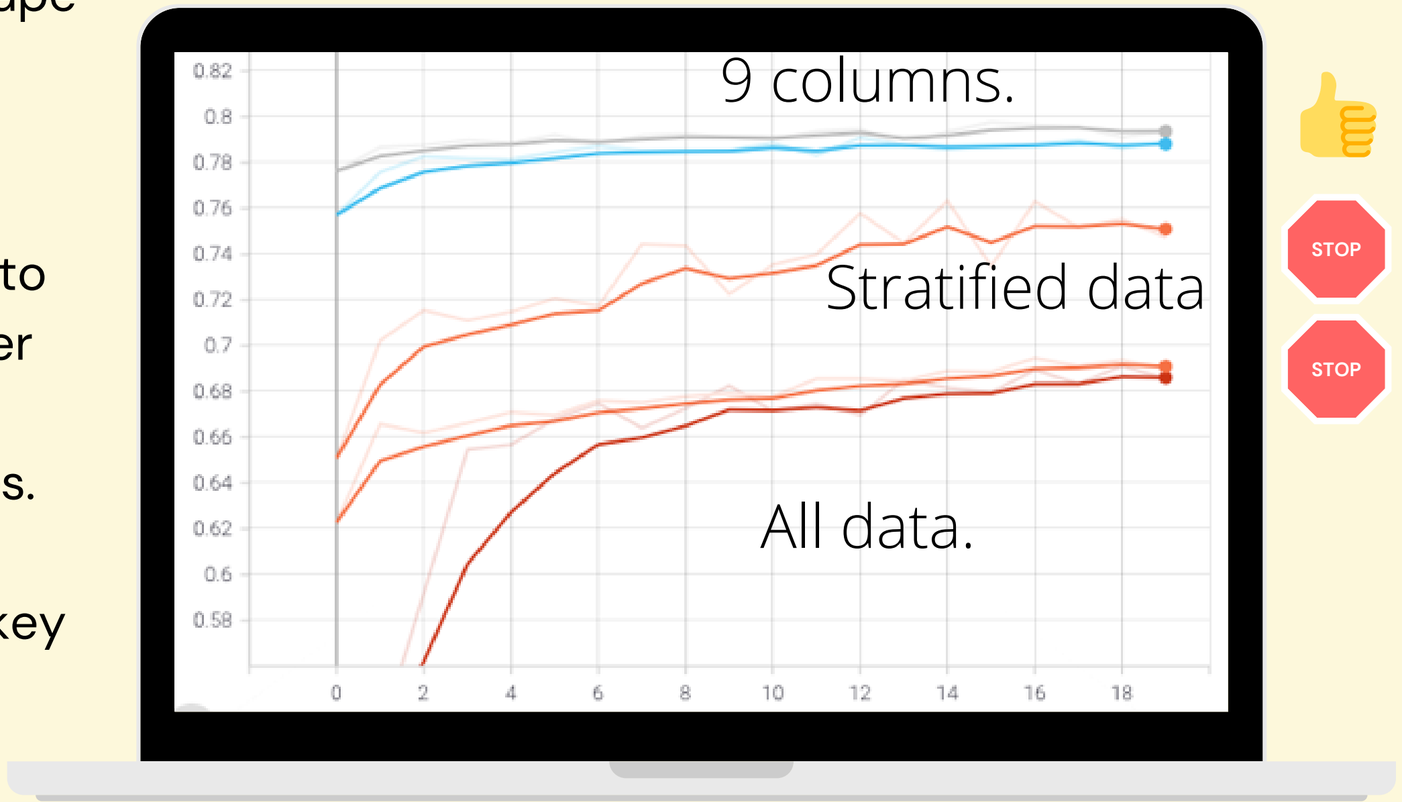
- Key data points:
  - Contract type.
  - Tenure.
  - Interent service type.

Data Processing

What do we do with the data?

DATA SIZE

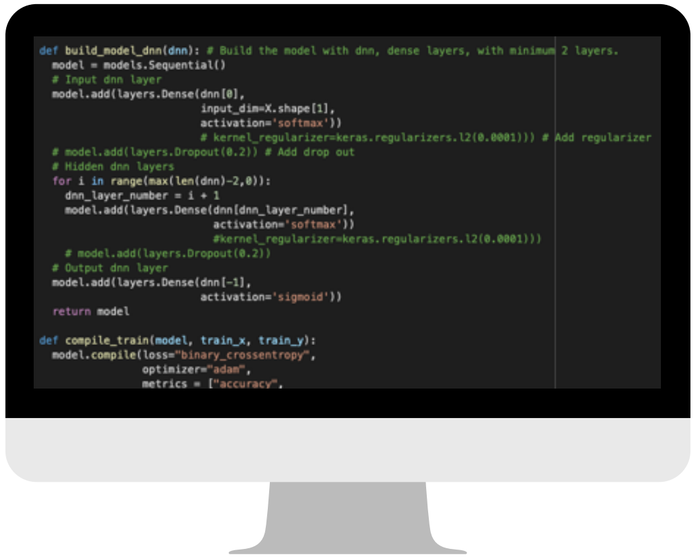
- Test sensitivity of result to data shape and processing.
  - Number of columns.
  - Stratify data to achieve higher balance of binary classes.
  - One hot encoding of key parameters.



Test Accuracy by Epoch

- Proceed with data set using.
  - 9 columns.
  - One hot encoding of two columns.
    - Interent service type "DSL"
    - Contract type "Month-to-Month"

Model



```
def build_model_dnn(dnn): # Build the model with dnn, dense layers, with minimum 2 layers.
  model = models.Sequential()
  # Input dnn layer
  model.add(layers.Dense(dnn[0],
                         input_dim=X.shape[1],
                         activation='softmax'))
                         # kernel_regularizer=keras.regularizers.l2(0.0001))) # Add regularizer
  # model.add(layers.Dropout(0.2)) # Add drop out
  # Hidden dnn layers
  for i in range(max(len(dnn)-2,0)):
    dnn_layer_number = i + 1
    model.add(layers.Dense(dnn[dnn_layer_number],
                           activation='softmax'))
                           #kernel_regularizer=keras.regularizers.l2(0.0001)))
    # model.add(layers.Dropout(0.2))
  # Output dnn layer
  model.add(layers.Dense(dnn[-1],
                         activation='sigmoid'))
  return model

def compile_train(model, train_x, train_y):
  model.compile(loss="binary_crossentropy",
                optimizer="adam",
                metrics = ["accuracy",
```

Build	Compile	Fit
Conventional DNN model. **Input:** DNN (32) Softmax **Hidden layer:** DNN (16) Softmax **Output layer:** DNN (1) Sigmoid	Extra metrics for manual calculations later. •Accuracy •False Negatives •True Negatives •False Positives •True Positives	Validation split = 25%. Epochs = 20 (Test data is 30%)

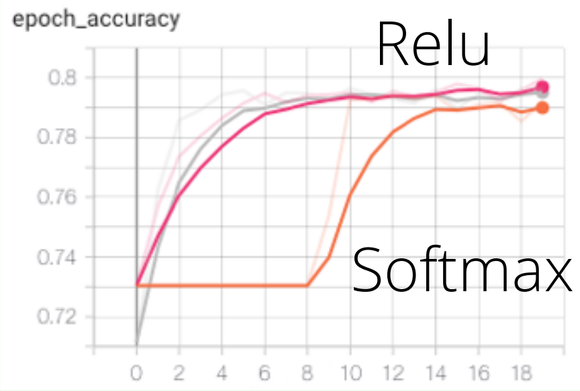
Model Tuning

Sensitivity analysis to improve model results.

- Very similar loss and accuracy results, but different accuracy for positive results.
  - Softmax
    - 67% identified.
  - Relu
    - 57%
- Decision point...

BUILD

- Number of layers.
- Size of layers.
- Output layer activation function.
- DNN activation functions:
  - Softmax vs relu



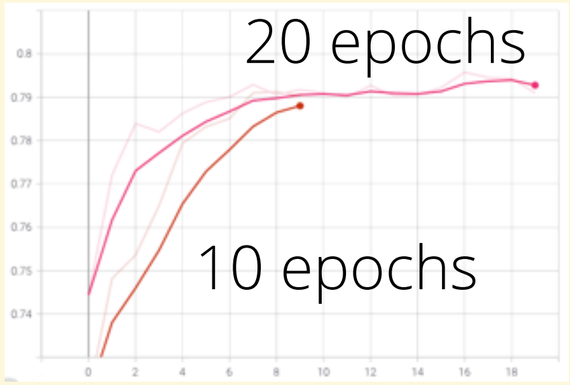
👍 Softmax or Relu?

COMPILE

- Optimiser.
  - SGD vs adam
  - SGD very poor result.
- Metrics.
  - False positive and negatives.
  - Use for calculating detection rate.
- Few useful changes.

FIT

- Epochs.
- Optimiser.
  - SGD vs adam
- Metrics.
  - False positive and negatives.



✋ Less epochs needed

Business Decision

What is more costly?

FALSE

Reduced margin on a customer who wouldn't have churned...

- Softmax with higher true positive rate.

Lose a customer who might have been retained...

- Relu with lower false positive rate.

Assume non-churn customers would move to newer deals with same provider anway, or are less likely to take action. Therefore select Softmax.

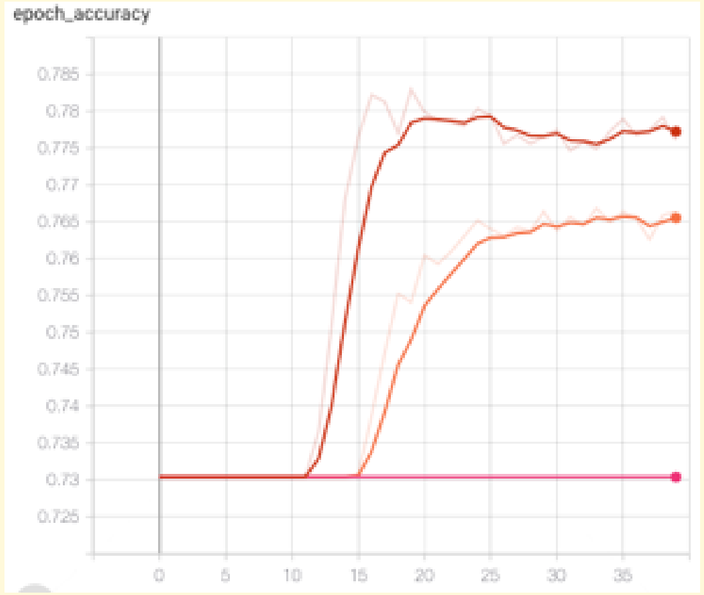
Final Tuning

Optimise against over and under fitting.

TUNE

Regularization

Applied regularization to attempt greater learning over more epochs.



Base line

L2

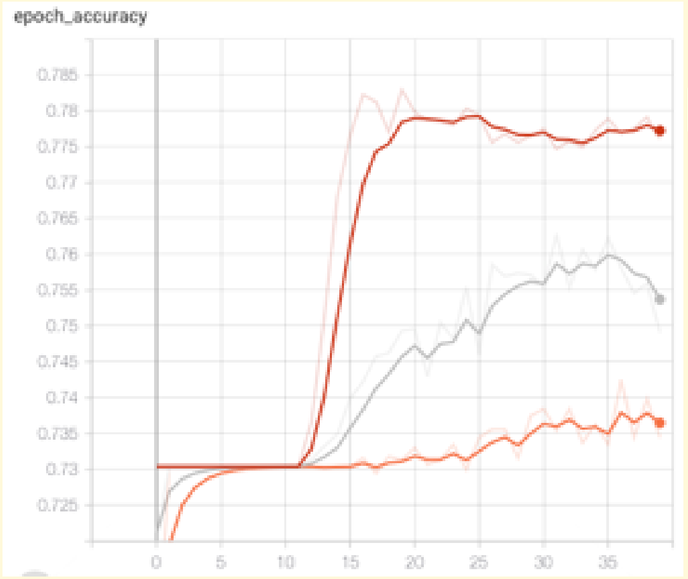
L1

Returned more biased toward False.

STOP

Drop Out

Applied drop out to attempt greater learning over more epochs.



Base line

D = 0.2

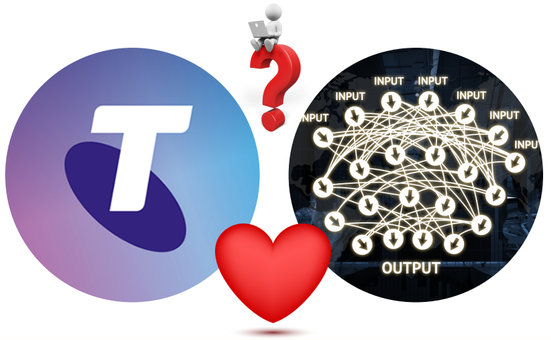
D = 0.5

Returned more biased toward False.

STOP

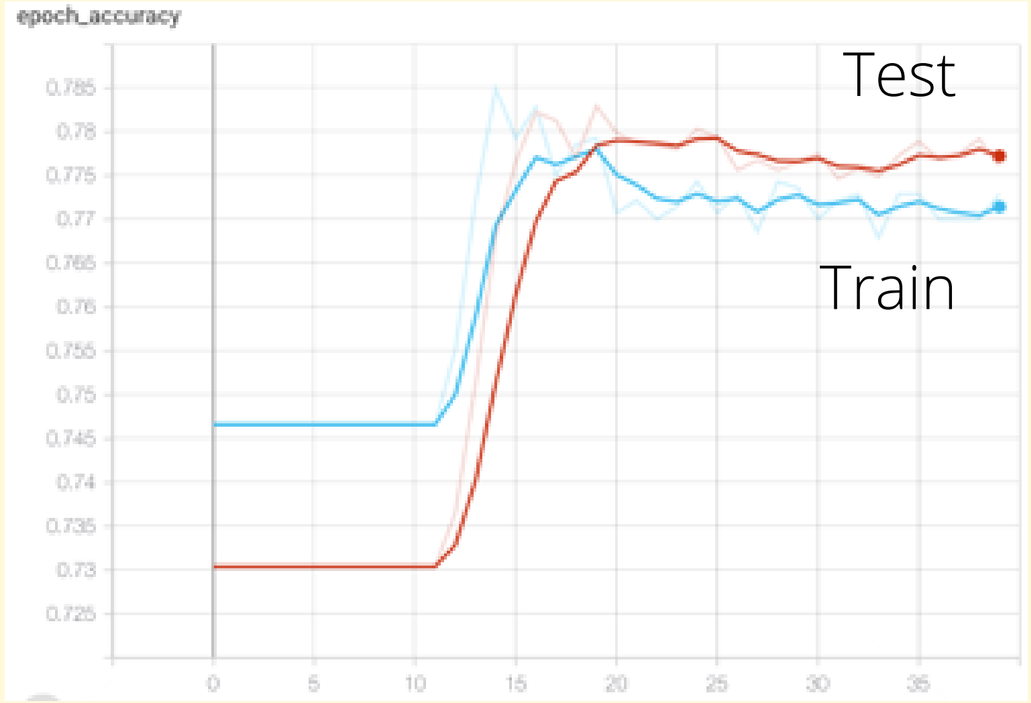
Performance

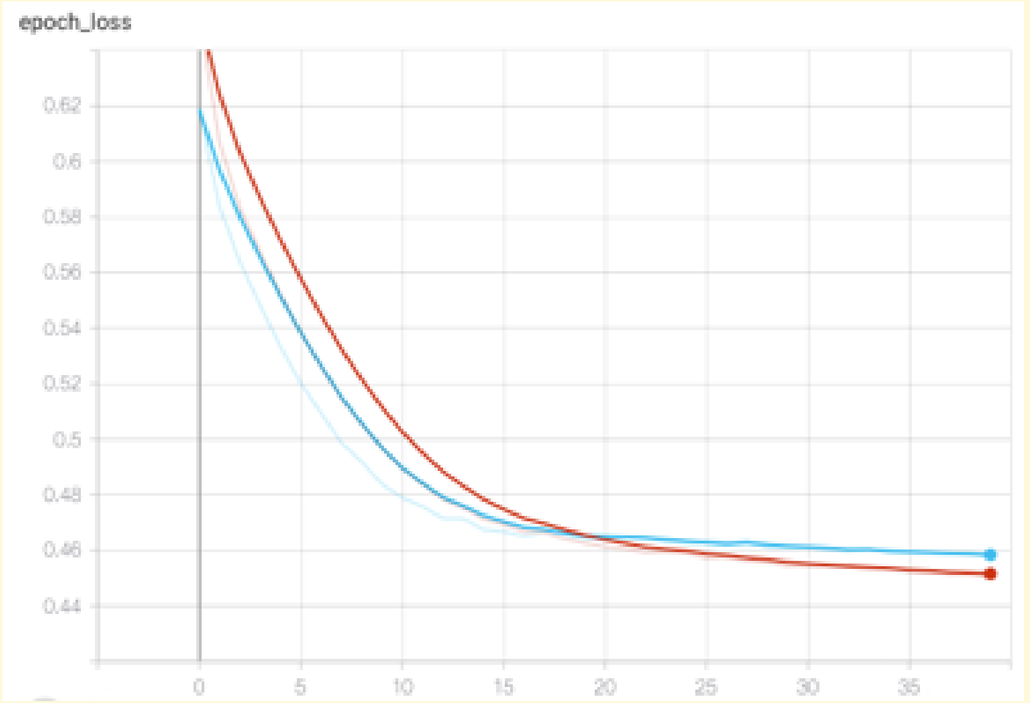
Problem solved?



Business problem solved with caveats.

- Best performing model could detect 61% of customers who would churn.
- However it gave false positives for 16% of customers who would not churn.





Next Steps

How to get even better?

- Collect other data with stronger correlation to customer status.
- Investigate models which return probability of customer churning so that a graded retention response can be given.

Lincoln Innes

19 December 2020

Thank you

&
Merry
Christmas