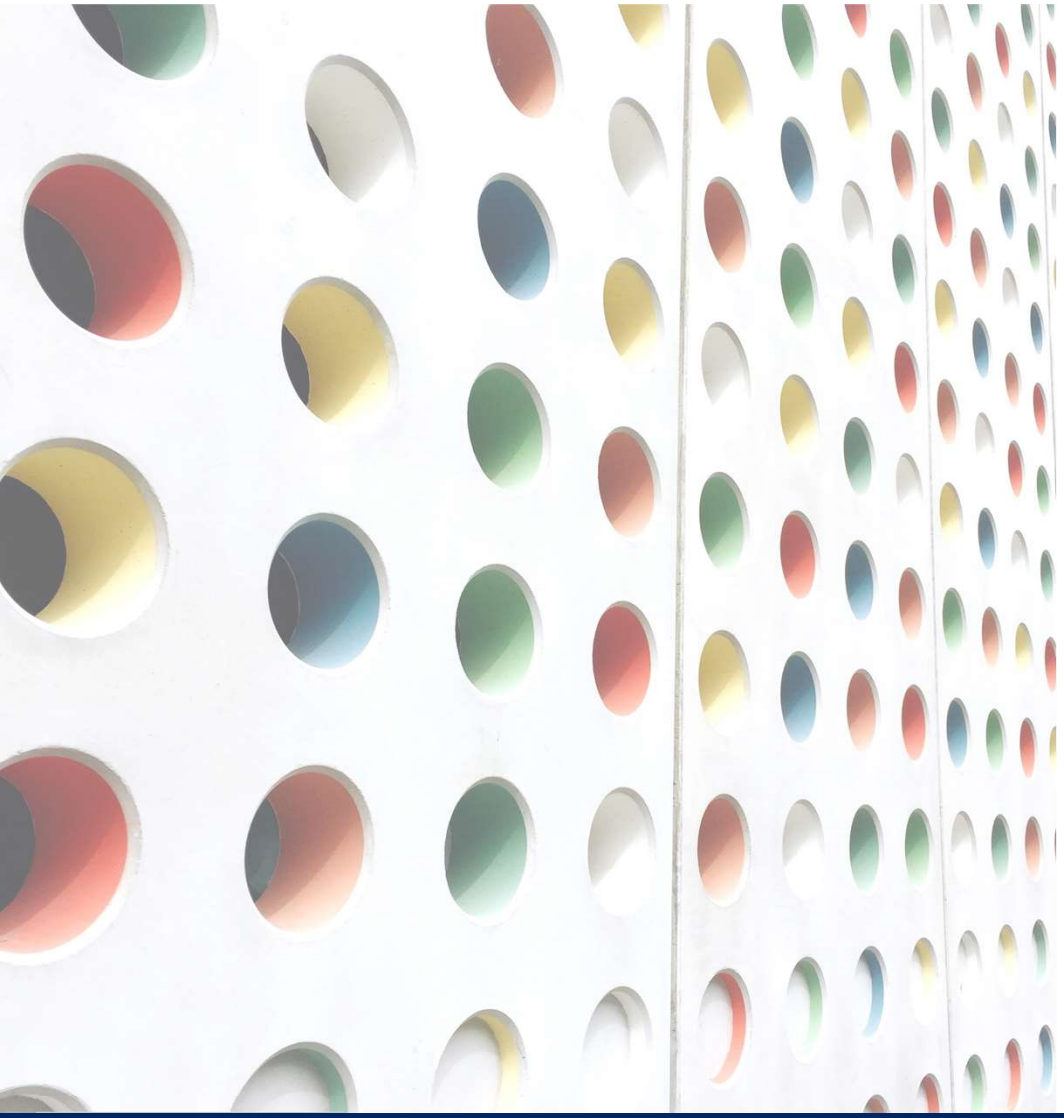


Neural Network Assignment

MYKA MILLER

Part 1 ~ Results

	----- Loss Metric (MSE) -----			
Model	Training Set	Validation Set	MSE on Test	Predicted Vaccination Rate for Delaware County, PA
Linear Model	0.51%	0.52%	0.67%	68.49%
Neural Network	0.75%	0.85%	0.95%	48.47%

Part 1 ~ Neural Network Design

The biggest positive impacts to my model were:

- Forming uniform layers
  - started out with all layers different, like in part 2,
    - saw much more improvement with all layers having the same amount of neurons
- Early Stopping
  - I didn't start any model without this, I'm sure it saved me time and overfit data
- Maintaining original learning rate of 0.001

```
#the same steps above can be built and saved in a single command with the same results
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=X_train.shape[1:]),
    keras.layers.Dense(15, activation="relu"),
    keras.layers.Dense(15, activation="relu"),
    keras.layers.Dense(15, activation="relu"),
    keras.layers.Dense(15, activation="relu"),
    keras.layers.Dense(1)
])
```

```
#After model is created, it needs to be compiled - this requires setting
#the loss function to mean_squared_error
model.compile(loss="mean_squared_error",
              optimizer=keras.optimizers.SGD(learning_rate=.001),
             )
```

```
#fit the model and capture the details of the fit to a variable called history
#note that validation data is dynamically allocated at 20% of the training data

early_stopping = keras.callbacks.EarlyStopping( monitor='val_loss', mode='min', patience=2)

history = model.fit(X_train,
                    y_train,
                    epochs=500,
                    validation_split=.2,
                    callbacks=[early_stopping]
                   )
```

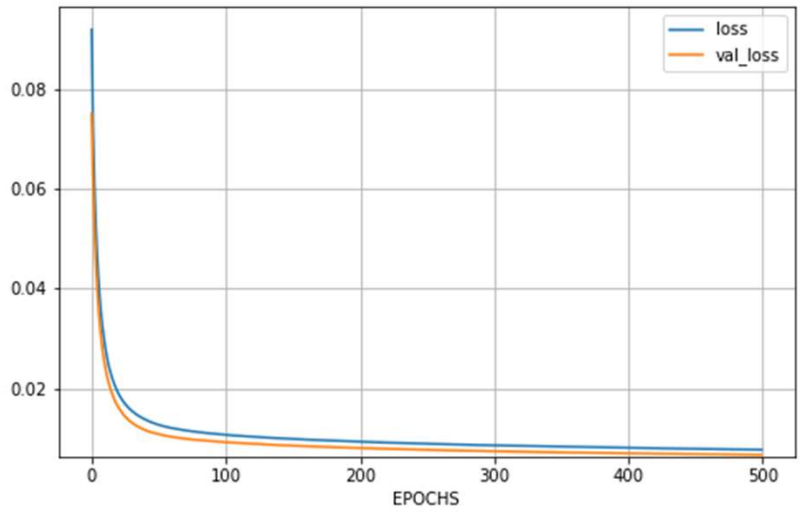
Part 1 ~ Neural Network Assessment

The model is underfitting the training data slightly

Both training and validation have reached a nice plateau

- Feel comfortable with the amount of learning the model has done

- Overall, this model had higher MSE scores than my traditional model. I think both regression models would benefit from further tuning before making a decision on which to use, though.



Part 2 ~ Results

---------- Accuracy Score ----------

Model	Training Set	Cross Validation on Training Set	Test Set
Logistic Model	68.34%	67.79%	68.46%
Neural Network	72.30%	71.78%	73.04%

Part 2 ~ Neural Network Design

The biggest positive impacts to my model were:

- The inclusion of dropout layers
  - Learning curves converged better and started to plateau
- Using Adam instead of SGD
  - Forming pyramid with neurons
    - started out with all layers the same, saw much more improvement with narrowing down
  - Early Stopping
    - I didn't start any model without this, I'm sure it saved me time and overfit data
  - Adjusting Learning Rate
    - 0.001 was still erratic, but 0.0001 had too much learning left. 0.0003 was a good middle ground

```
#the same steps above can be built and saved in a single command with the same results
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=X_train.shape [1:]),
    keras.layers.Dense(60, activation="relu"),
    keras.layers.Dropout(.2),
    keras.layers.Dense(60, activation="relu"),
    keras.layers.Dropout(.2),
    keras.layers.Dense(50, activation="relu"),
    keras.layers.Dropout(.2),
    keras.layers.Dense(25, activation="relu"),
    keras.layers.Dropout(.2),
    keras.layers.Dense(1, activation="sigmoid")
])
```

```
model.compile(loss='binary_crossentropy',
              optimizer=keras.optimizers.Adam(learning_rate=.0003),
              metrics=['accuracy']
             )
```

```
#fit the model and capture the details of the fit to a variable called history
#validation data will be created dynamically as 20% of the training data set

early_stopping = keras.callbacks.EarlyStopping( monitor='val_loss', mode='min', patience=2)

history = model.fit(X_train,
                    y_train,
                    epochs=500,
                    validation_split=.2,
                    callbacks=[early_stopping]
                   )
```

Part 2 ~ Neural Network Assessment

Model starts off underfitting training data, but as it learns and finishes, it is slightly overfitting training

Both validation and training are converging towards the end of the learning

Slightly erratic behavior for both sets of curves

- May benefit from adjusting the learning rate a little more

Curves haven't quite finished plateauing, still a little more for the model to learn

Overall, I am satisfied with this model and would consider using it over the traditional models from the past module..