Section VI: References

Acharya, U. R., Fernandes, S. L., WeiKoh, J. E., Ciaccio, E. J., Fabell, M. K., Tanik, U. J., Rajinikanth, V., & Yeong, C. H. (2019). Automated detection of Alzheimer's disease using brain mri images– a study with various feature extraction techniques. Journal of Medical Systems, 43(9). https://doi.org/10.1007/s10916-019-1428-9

Bellio, M., Oxtoby, N. P., Walker, Z., Henley, S., Ribbens, A., Blandford, A., Alexander, D. C., & Yong, K. X. (2020). Analyzing large alzheimer's disease cognitive datasets: Considerations and challenges. Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring, 12(1). https://doi.org/10.1002/dad2.12135

Brooks, A. (2021, December 6). How much does an MRI cost without insurance in 2021? Mira. Retrieved January 27, 2022, from https://www.talktomira.com/post/how-much-does-an-mri-cost-without-insurance-in-2021

Bunis, D. (2019, September 12). Numbers of Americans without health insurance grows. AARP. Retrieved October 25, 2021, from https://www.aarp.org/health/health-insurance/info-2019/older-americans-not-insured.html

Centers for Disease Control and Prevention. (2020, October 26). What is alzheimer's disease? Centers for Disease Control and Prevention. Retrieved October 24, 2021, from https://www.cdc.gov/aging/aginginfo/alzheimers.htm

Cardenas VA, Chao LL, Studholme C, Yaffe K, Miller BL, Madison C, Buckley ST, Mungas D, Schuff N, Weiner MW. Brain atrophy associated with baseline and longitudinal measures of cognition. Neurobiol Aging. 2011 Apr;32(4):572-80. doi:

10.1016/j.neurobiolaging.2009.04.011. Epub 2009 May 14. PMID: 19446370; PMCID: PMC2891686.

Christ, B. U., Combrinck, M. I., & Thomas, K. G. (2018). Both reaction time and accuracy measures of intraindividual variability predict cognitive performance in Alzheimer's disease. Frontiers in Human Neuroscience, 12. https://doi.org/10.3389/fnhum.2018.00124

Harrison, O. (2019, July 14). Machine learning basics with the k-nearest Neighbors ALGORITHM. Medium. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

Maroco, J., Silva, D., Rodrigues, A., Guerreiro, M., Santana, I., & de Mendonça, A. (2011). Data mining methods in the prediction of dementia: A real-data comparison of the accuracy, sensitivity and specificity of linear discriminant analysis, logistic regression, neural networks, support vector machines, classification trees and random forests. BMC Research Notes, 4(1). https://doi.org/10.1186/1756-0500-4-299

Nawaz, A., Anwar, S. M., Liaqat, R., Iqbal, J., Bagci, U., & Majid, M. (2020). Deep Convolutional Neural Network based Classification of Alzheimer's Disease using MRI Data. 2020 IEEE 23rd International Multitopic Conference (INMIC). https://doi.org/10.1109/inmic50486.2020.9318172

Schaffer, J. D., & Chiofolo, C. M. (2016, June 14). Methods and systems for identifying patients with mild cognitive impairment at risk of converting to alzheimer's. (US 9367817 B2), United States Patent, https://www.freepatentsonline.com/9367817.pdf

Section VII: Appendices

Appendix A: Limitations and Assumptions

Limitations:

1. Unable to test on human participants
2. Trained with a machine learning method that did not use a deep learning algorithm
3. The database had a size of about 9,000 data points after pre-processing.

Assumptions:

1. The Mini-Mental Score provides a correlation to Alzheimer's status as researched and stated by the ADNI database.
2. Genetic factors do not play a role in further progression of Alzheimer's disease.

Appendix B: *Sample of Data Set used in the Model*

RID	VISCODE2	ADNI_MEM	ADNI_EF	AGE	GENDER	EDUCATION	MMSCORE	ADDIAGNOSIS
2	0	0.486	0.091	74	1	16	28	0
2	6	0.581	0.088	75	1	16	28	0
2	36	0.405	0.088	77	1	16	29	0
2	60	0.352	-0.294	79	1	16	28	0
2	72	0.35	0.343	80	1	16	23	1
2	84	0.428	-0.274	81	1	16	24	1
2	96	0.325	-0.79	82	1	16	25	0
2	108	0.089	-0.75	83	1	16	28	0
2	120	0.384	-0.609	84	1	16	28	0
2	132	0.164	-0.711	85	1	18	24	1

Appendix C: *LSTM Code*

```
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM

import tensorflow as tf
from google.colab import drive
drive.mount("/content/drive")
tf.config.list_physical_devices('GPU')

dataset = read_csv("/content/drive/MyDrive/Datasets/Dataset.csv")
dataset.head()

values = dataset.values
# specify columns to plot
```

```
groups = [1, 2, 3, 5, 6]
i = 1
# plot each column
pyplot.figure()
for group in groups:
    pyplot.subplot(len(groups), 1, i)
    pyplot.plot(values[:, group])
    pyplot.title(dataset.columns[group], y=0.5, loc='right')
    i += 1
pyplot.show()

# convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
                cols.append(df.shift(i))
                names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]
    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
                cols.append(df.shift(-i))
```

```
47.             if i == 0:
48.                     names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
49.             else:
50.                     names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
51.     # put it all together
52.     agg = concat(cols, axis=1)
53.     agg.columns = names
54.     # drop rows with NaN values
55.     if dropnan:
56.             agg.dropna(inplace=True)
57.     return agg
58.
59. # load dataset
60. dataset = read_csv("/content/drive/MyDrive/Datasets/Dataset.csv", header = 0,index_col
    = 0)
61. values = dataset.values
62. # integer encode direction
63. encoder = LabelEncoder()
64. values[:,4] = encoder.fit_transform(values[:,4])
65. # ensure all data is float
66. values = values.astype('float32')
67. # normalize features
68. scaler = MinMaxScaler(feature_range=(0, 1))
```

```
scaled = scaler.fit_transform(values)
# frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
# drop columns we don't want to predict
reframed.drop(reframed.columns[[6,7,8,9,10]], axis=1, inplace=True)
print(reframed.head())

# split into train and test sets
values = reframed.values
n_train_days = 300
train = values[:n_train_days, :]
test = values[n_train_days:, :]
# split into input and outputs
train_X, train_y = train[:, :-1], train[:, -1]
test_X, test_y = test[:, :-1], test[:, -1]
# reshape input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))
print(train_X.shape, train_y.shape, test_X.shape, test_y.shape)

# design network
model = Sequential()
model.add(LSTM(8, input_shape=(train_X.shape[1], train_X.shape[2])))
```

```
model.add(Dense(1))
model.compile(loss='mae', optimizer='adam', metrics=['accuracy'])
# fit network
history = model.fit(train_X, train_y, epochs=8, batch_size=72, validation_data=(test_X,
    test_y), verbose=2, shuffle=False)
# plot history
pyplot.plot(history.history['loss'], label='train')
pyplot.plot(history.history['val_loss'], label='test')
pyplot.title('Training and Validation loss')
 pyplot.xlabel('Epochs')
 pyplot.ylabel('Loss')
 pyplot.legend()
 pyplot.show()

 pyplot.plot(history.history['accuracy'], label='train')
 pyplot.plot(history.history['val_accuracy'], label='test')
 pyplot.title('Training and Validation Accuracy')
 pyplot.xlabel('Epochs')
 pyplot.ylabel('Accuracy')
 pyplot.legend()
 pyplot.show()

 # make a prediction
```

```
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))
# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,0]
# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:,0]
# calculate RMSE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)

loss_train = history.history['accuracy']
loss_val = history.history['val_accuracy']
epochs = range(1,11)
pyplot.plot(epochs, loss_train, 'g', label='Training accuracy')
pyplot.plot(epochs, loss_val, 'b', label='validation accuracy')
pyplot.title('Training and Validation accuracy')
pyplot.xlabel('Epochs')
pyplot.ylabel('Accuracy')
```

```
pyplot.legend()
pyplot.show()
```

Appendix D: ***Code for Optimization Method***

```
def optimizeParameters(df):
    acc = 0
    n = 0
    drop = 0;
    for d in range(30,70):
      df = tempdf
      for y in range(len(df)):
        if(df.MMSCORE[y] >= 27 and random.randint(1,100) <= d):
          df.drop(y,inplace = True)
      X = df[["ADNI_MEM", "ADNI_EF"]]
      y = df[["MMSCORE"]]
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
      for x in range(15,80):
        knn_model = KNeighborsRegressor(n_neighbors=x)
        knn_model.fit(X_train, y_train)
        test_preds = knn_model.predict(X_test)
        if (metrics.r2_score(y_test,test_preds) > acc):
          acc = metrics.r2_score(y_test,test_preds)
```

```
19.             n = x
20.             drop = d
21.         print(acc)
22.         print(n)
23.         print(drop)
24.
```

Appendix E: ***Decision Matrix Comparing Different Machine Learning Methods***

Criteria	Max Points	Multi Variate Regression	LSTM + kNN
1. Predicts next value for an inputted time step (required)	10	6	10
2. Prediction Accuracy above 75%	9	4	7
3. Classifies Alzheimer's using MM	8	7	8

score with RMSE below 4			
Total	27	17	25
Percent	100%	63%	93%

- Multi Variate Regression: A model that simply uses a linear regression model to create parametric equations to produce a predicted value; not as accurate to training set
- LSTM + kNN: A model that uses a neural network to predict future cognitive values and then classify using the kNN