DSC 383W Mini-Project report: Human Activity Recognition

Tanmay Thakkar

Introduction:

Human Activity Recognition (HAR) is certainly a very popular research area in computer vision and human-computer interaction. HAR aims to successfully classify different human activities based on a series of observations obtained from sensors that are affected by human movement[1]. About 45.12% of the world's population owns a smartphone[2] and thus, the embedded sensors within smartphones make it now possible to collect a vast amount of data on human activities.

The two sensors, accelerometer, and gyroscope are commonly found in most of the smartphones and have specific tasks. Accelerometers are essentially used to detect the orientation of the phone and measure the rate of change of speed of a movement[3]. On the other hand, gyroscopes add another level of precision to the information provided by the accelerometers by also sensing angular rotational velocity and acceleration[4].

After obtaining the readings from these sensors I first conducted an exploratory analysis and then implemented multiple classification models to classify the observations into different human activities performed by the individuals in the experiment, with a certain degree of accuracy. My analysis suggested that from the large array of features in the dataset only a few are sufficient to obtain a significantly high classification accuracy.

Methods:

Data Collection

For my analysis, I used a dataset consisting of signals from a smartphone carried by 30 individuals within the age range of 19-48 years performing six different activities. Six different singles were captured (3-axial linear acceleration signals from the accelerometer and 3-axial angular velocity signals from the gyroscope) at a constant rate of 50Hz. The data set consisted of a total of 10,299 observations and 561 features which were specifically engineered from raw accelerometer and gyroscope signals. I also had access to training and testing data where the training set included 7352 observations and the testing set included the remaining 2947 observations. The data was provided by Professor Ajay Anand as part of a data science capstone course.

Exploratory Analysis

An exploratory analysis was conducted in order to make sense of the given data and in an attempt to gather as many insights as possible before implementing the classification models. I began my analysis by checking for any null values in the dataset. Luckily the dataset consisted of zero null values and thus I moved on to verify if there existed a class imbalance problem.

Class imbalance occurs when the class labels in the dataset are not of approximately equal size. This could be a serious problem as large class imbalances in training data could lead to biased results when implementing machine learning algorithm leading to poor predictive performance. Looking at Figure 1, it is clear that the share of each class label (i.e. human activity) is of the same size and thus, the dataset is perfectly balanced and ready to use as it is. Lastly, due to a

large number of features, I decided to compute the correlation amount the attributes in an attempt to eliminate the ones that were highly correlated as they do not contribute towards adding any predictive power. Moreover, narrowing down a high dimensional could also potentially alleviate the problem of the classification model overfitting on the training data which could, in turn, lead to higher classification accuracy on the test data. I set the threshold of correlation to 0.9 and thus pairs of features with correlation coefficients of greater than or equal to the threshold were discarded from the training and testing datasets. 416 highly correlated were eliminated from the dataset which led to a significant reduction in dimensionality.

Figure 1. Distribution of class labels

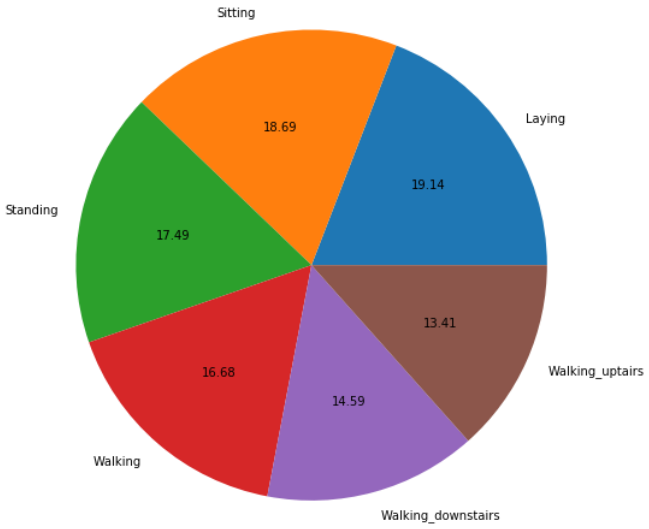
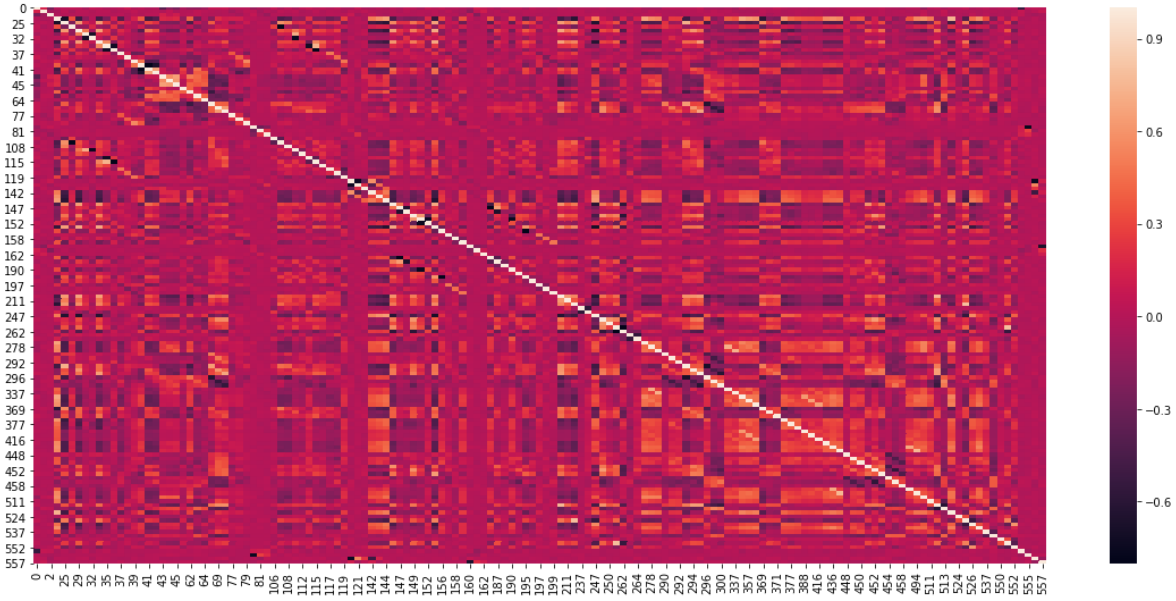


Figure 2. Correlation amongst the set of attributes



Classification Models:

Support Vector Machines (SVM)

The basic idea of an SVM is that given labeled training data the algorithm finds optimal linear or non-linear hyperplanes to separate the classes. The goal of the algorithm is to maximize the separability of the classes to minimize misclassification[5]. SVM does a great job of controlling the complexity of high dimensional datasets like ours by its regularization capabilities. In other words, for a non-linearly separable dataset (i.e. a dataset with a large number of attributes), the minimizing misclassification condition is relaxed to accurately categorize the new data point.

In my analysis, I fit two different versions of SVM's where the only difference is in the type of kernel. As I was unsure if the data was linearly separable, I started by choosing the kernel as linear followed by a polynomial kernel in order to compare how each of them performed on test data. Based on the requirement of objective one, both these models were first trained and run on all the 561 features which did include highly correlated variables as well. Further, I fit the linear SVM again after performing feature selection as described below.

K-Nearest Neighbor (k-NN)

k-NN is a simple and yet widely used non-parametric classification algorithm as unlike SVMs, it does not make any assumption about the underlying data[6]. In other words, all data points are used for training the algorithm while new observations are classified. The basic idea of k-NN is that it calculates the distance or similarity of a new data point to all other training data points and then tries to find k closest or most similar neighbors. "k" is a user-defined integer which is used to determine the number of neighbors that need to be found by the algorithm. Lastly, the new data point is assigned to a class label in which most of the k neighbors exist.

Deciding on an ideal value for k is an important yet convoluted task. Low values of k can make the model less robust to noise whereas high values of k could mean the inclusion of irrelevant class labels. Both of these issues could highly impact the performance of the algorithm. As objective one required using all the 561 features, there was no scope for filtering any noise data points. Thus, I randomly picked k as 7 to get a rough idea as to how k-NN was performing.

Feature Selection Process:

Recursive Feature Elimination with Cross Validation (RFECV)

Recursive feature elimination (RFE) is essentially a feature elimination technique that works by building a model on all of the predictors and weighs each one of them by computing an importance score[7]. The attributes with the lowest importance scores are removed and the model is re-built, and the importance scores are computed again. When using simple RFE the user must specify the number of features to select and thus the recursive process terminates when the optimal number of user-specified attributes are selected to train the final model.

To make feature elimination more robust, I decided to first use k-fold cross validation along with RFE. Cross-validation is a process that adds another level of robustness to the simple train-test split of the dataset. It begins by dividing the dataset into "k" user-specified bins and runs learning experiments k times. For each experiment, one of the bins is treated as the testing set

and the other k-1 bins are the training set and then the model's performance on the testing set is calculated. Eventually, the average of k different testing performances is computed[8].

In the case of RFECV, the user is not required to input the number of features that need to be selected but rather only the value for k needs to be specified to partition the dataset along with the estimator and step size and scoring metric. I choose the linear SVM as my estimator, choose stratified k-fold with k=5 as the method of cross-validation, set the step size (i.e. the number of features to remove at each iteration) equal to 1 and the scoring metric as accuracy score.

Results:

As part of objective one, I first fit the linear SVM classifier followed by the polynomial SVM on all of the 561 features in the dataset. The accuracy scores of the models on the test data were 96.4% and 90.7% respectively. As SVM's generally perform well on high-dimensional data, it was not surprising to get reasonably high accuracy scores. Moreover, looking at the significant difference in the two scores it is clear that the dataset is linearly separable.

Figure 3. Confusion matrix for linear SVM

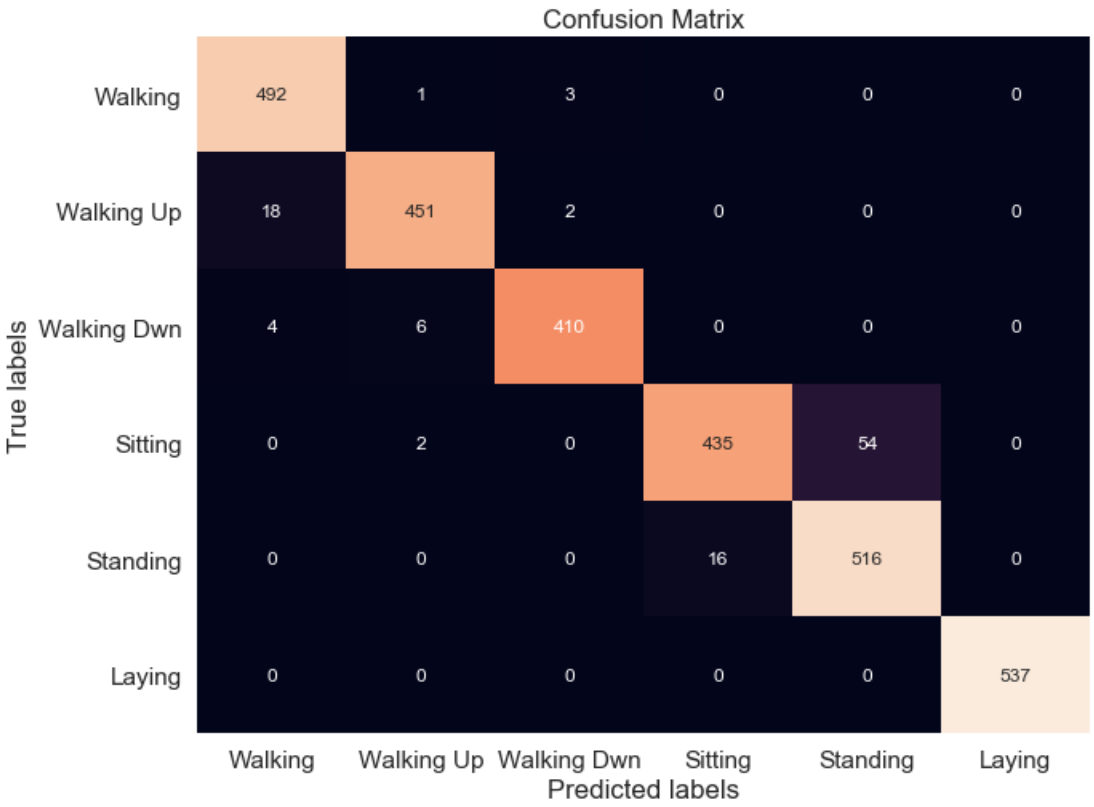
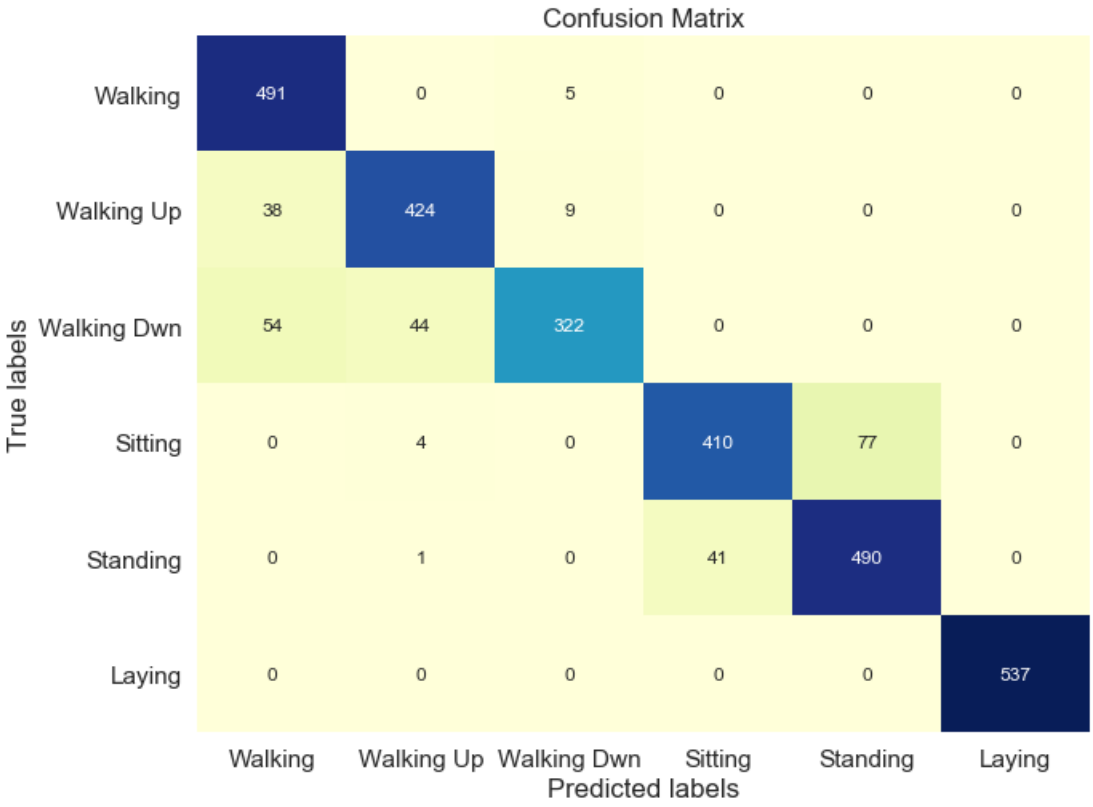


Figure 4. Confusion matrix for polynomial SVM



For a baseline comparative analysis, I fit the k-NN classifier on the complete feature set as well. The resulting accuracy score of the model was 90.3% which is marginally lower than the polynomial SVM. As expected, the predictive power of k-NN is much lower than that of linear SVM however, I was surprised by the fact that k-NN had a reasonably high accuracy score despite high dimensionality. This could probably explain the importance of domain-specific knowledge when generating the feature set from the raw data.

For objective two I performed recursive feature elimination on the remaining 145 features that were in the dataset after removing highly correlated variables. Running RFE on a model can be very expensive and thus it was necessary to narrow down the feature set by removing attributes that provide the same information. Performing RFECV on linear SVM along with other specifications mentioned earlier, resulting in a total of 122 optimal features which were then used to test the classification accuracy. The training and the testing sets were transformed to only contain these features and the resulting accuracy score was 94.4%. This score is slightly lower than the score obtained by fitting SVM using all the 561 features, however, feature elimination is still useful. This is because it shows that choosing only a small, but an optimal subset of the entire feature set could still lead to a very high classification accuracy on test data and also boosts the time required to train the SVM.

Figure 5. Confusion matrix for k-NN

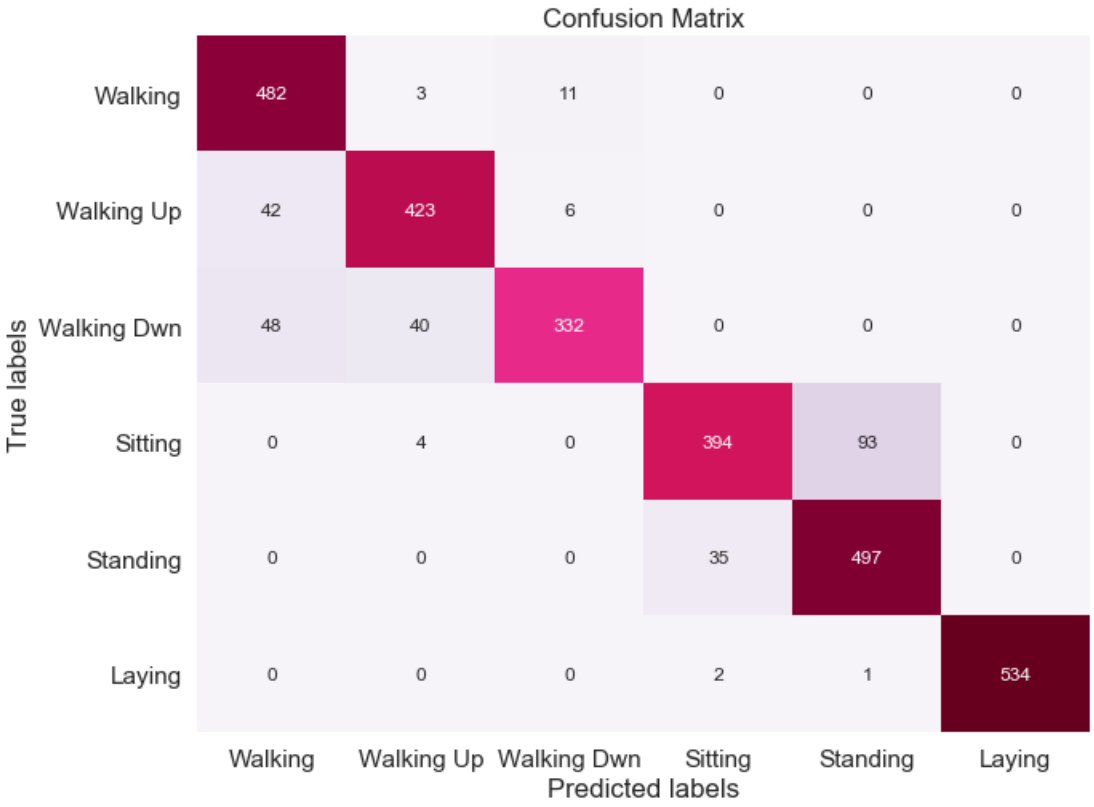
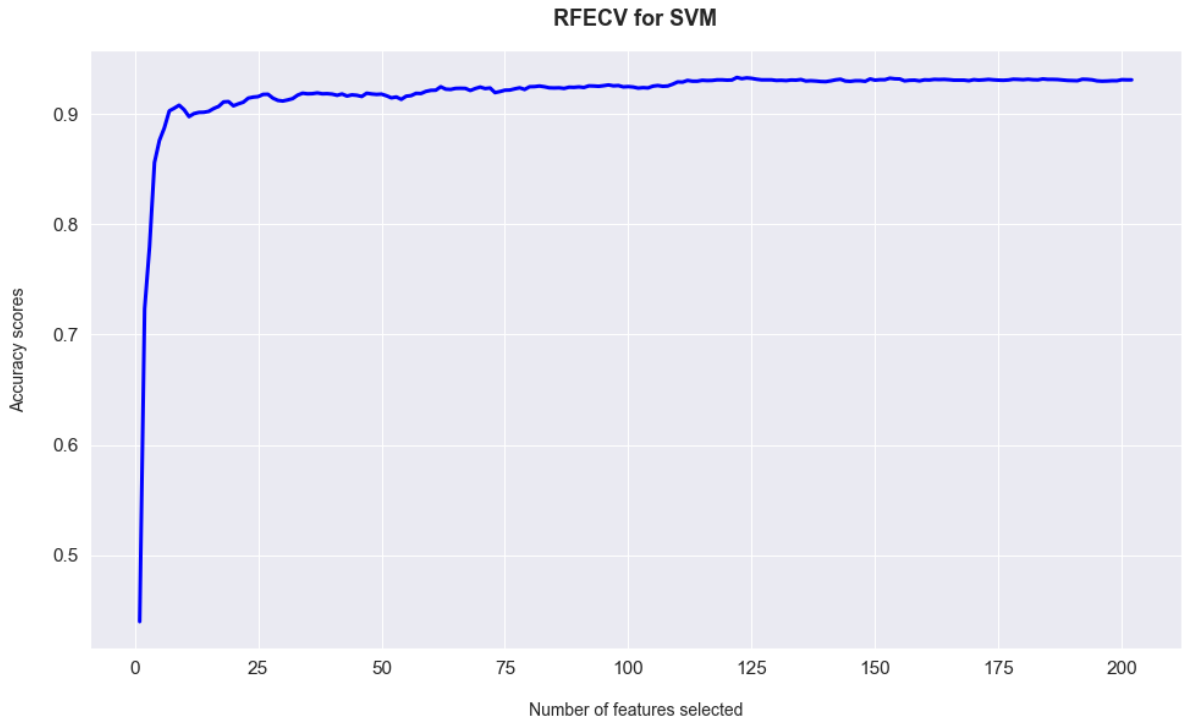


Figure 6. Accuracy scores for the number of features used by RFECV



As RFECV does not allow for specification of the exact number of features to select, I decided to run RFE for linear SVM without cross-validation on the 122 optimal features produced by RFECV. The aim was to find out the minimum features required to attain an 80% and 90% testing accuracy. Surprisingly only 4 optimal attributes were sufficient to attain an 83% accuracy whereas a 90% accuracy was achieved only by having at least 35 features.

Conclusion:

Accuracy scores of the tested models are given in Table 1 below. While linear SVM has the best classification accuracy as seen in Table 1, the k-NN classifier also performed much better than expected. k-NN requires points to be close to each other in every dimension to perform accurate classification, however, as new dimensions get added to a dataset it becomes harder for two points to be close to each other[9]. But in this case of the HAR dataset, it seemed that there were enough observations to maintain the density of the dataset even when the features increase and thus k-NN did not lose much of its predictive power. Performing feature elimination did not help with improving the accuracy of SVM but as mentioned before it did give some valuable insight and improved runtime. Moreover, altering the step size when performing RFECV could potentially improve performance as it would get rid of worse features earlier and consequently strengthen the remaining features. This also helps to increase the speed of feature elimination of high dimensional datasets.

Table1. Accuracy scores of tested models

Model	Accuracy Score
Linear SVM classifier without feature selection	96.4%
Polynomial SVM classifier without feature selection	90.7%
k-NN classifier without feature selection	90.3%
Linear SVM classifier after RFECV	94.4%

References:

1. Chin Ann, Ong & Lau, Bee. (2015). Human activity recognition: A review. Proceedings - 4th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2014. 389-393. 10.1109/ICCSCE.2014.7072750.
2. Turner, Ash. “1 Billion More Phones Than People In The World! BankMyCell.” *BankMyCell*, BankMyCell, 23 Jan. 2020, www.bankmycell.com/blog/how-many-phones-are-in-the-world.
3. “Accelerometers: What They Are & How They Work.” LiveScience, Purch, www.livescience.com/40102-accelerometers.html.
4. “Accelerometer vs. Gyroscope: What's the Difference?” LiveScience, Purch, www.livescience.com/40103-accelerometer-vs-gyroscope.html.
5. Tan, Steinbach, Karpatne, Kumar: Introduction to Data Mining second edition, pg 276

6. Chakure, Afroz. “K-Nearest Neighbors (KNN) Algorithm.” *Medium*, Towards Data Science, 1 Nov. 2019, towardsdatascience.com/k-nearest-neighbors-knn-algorithm-bd375d14eec7.
7. Rade, Dario. “Feature Selection in Python - Recursive Feature Elimination.” *Medium*, Towards Data Science, 2 Sept. 2019, towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15.
8. Brownlee, Jason. “A Gentle Introduction to k-Fold Cross-Validation.” *Machine Learning Mastery*, 8 Aug. 2019, machinelearningmastery.com/k-fold-cross-validation/.
9. Grant, Peter. “k-Nearest Neighbors and the Curse of Dimensionality.” *Medium*, Towards Data Science, 24 July 2019, towardsdatascience.com/k-nearest-neighbors-and-the-curse-of-dimensionality-e39d10a6105d.