Encouraging Anti-Diabetic Lifestyles in New Mexico Communities

New Mexico Supercomputing Challenge

Final Report

April 8, 2020

Team Number: 17

School: Grady High School

Team Members:

- **Alexis Brandsma** (alexisbrandsma@gmail.com) (575 231-9062)
- **Erynn Vetterly** (erynn.vetterly@gmail.com) (575 760-7601)
- **Tristen Pool** (tristenpool89@gmail.com) (575 760-1972)

Teacher:

- **Leah Lee** (llee@gradyschool.com)

Project Mentor:

- **Alan Daugherty** (adaugherty@melroseschools.org)

Table of Contents

Encouraging Anti-Diabetic Lifestyles in New Mexico Communities

Executive Summary:

Diabetes is a chronic disease that causes billions of dollars in economic losses every year. It consists of two main types: type 1 and type 2. Type 2 diabetes is the project's focus because it is inherently preventable. The cause of type 1 diabetes is not yet known, and there is not a proven way to prevent it [3]. 95 percent of diabetes cases are type 2. We aim to identify county demographic factors that correlate with diabetes rates and illustrate this with a program that is capable of accurately predicting a community's diabetes rate, given the appropriate data. First, we analyzed variables by making scatter plots using Python to extract .csv data. The variables that showed a correlation with the diabetes rates were inputted into our neural network (DANN). A neural network is an effective method to analyze variables because diabetes rates are complex, and the result of many different factors, our team needed a non-linear method to handle the demographic data properly. Mellitus is our project's central creation. It can accurately model a town, its demographics, and its hypothetical diabetes rate. Its vibrant interface is built-in Netlogo 6.1.1. An essential element to its performance is the implementation of DANN with the py extension. Mellitus and DANN can work together to make calculations accurately and effectively exhibit the results. Through this method, we identified five main variables that significantly impact diabetes rates: percent of American Indian and Alaska Native, poverty, education, commute time, and health insurance. Mellitus has proved to be able to consistently predict accurate diabetes rates for New Mexico counties using data from these variables.

Introduction:

Diabetes:

Diabetes is a chronic disease that affects how the pancreas functions. The pancreas can either be damaged by the immune system, which will cause a person to get type 1 diabetes. Or, the pancreas can't produce enough insulin for the body, which will result in a person contracting type 2 diabetes. Type 1 diabetes is less common than type 2. A person with type 1 diabetes will have to inject themselves with insulin, which would make them insulin-dependent [19]. The person would have to monitor their glucose levels as well. As for type 2 diabetics, they will have to take medication, and they would also have to monitor their glucose levels. If the body is having issues producing insulin, the person has to take an injection of medication that encourages the pancreas to operate properly. Some factors that can contribute to type 2 diabetes are an unhealthy diet, not being active, being overweight, and even genetics. About 95 percent of diabetes cases are type 2 [13]. Type 2 is reasonably prevented and is the focus of this project. The factors that can contribute to type 1 diabetes are genetics,

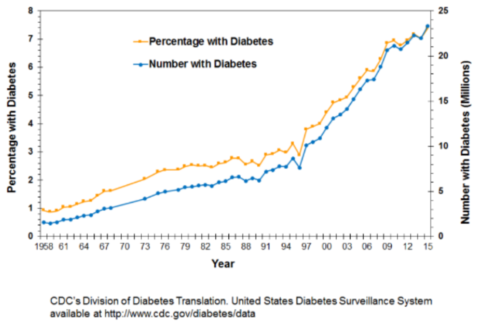


Figure 1- Number and percentage of U.S. population with diagnosed diabetes (1958 – 2015)

and a virus that tells the immune system to attack the pancreas. Type 2 diabetes is reversible through diet changes and weight loss and can result in a person's glucose levels returning to normal. Both types of diabetes have become increasingly common across the United States. Unlike type 2 diabetes, type 1 diabetes is neither reversible nor preventable.

Impact of Diabetes:

Diabetes has a consequential impact on New Mexico's health and economy. It affects the body's systems, which makes one's immune system weaker than in individuals without diabetes. Type 2 diabetes raises the risk of cardiovascular problems. These are just two examples of the tremendous economic losses that are caused by diabetes. In 2017, approximately 327 billion dollars were spent on diabetes-related costs such as buying insulin, a pump, medical supplies, medications, and doctor's visits in the U.S alone [11]. This cost does not include the loss of productivity from diabetes. Also, long term diabetes that is not treated properly can cause health problems as well. It can damage blood vessels, which can cause problems in the kidneys, eyes, and feet. In New Mexico, more than 14.1 percent of the population suffers from diabetes [12].

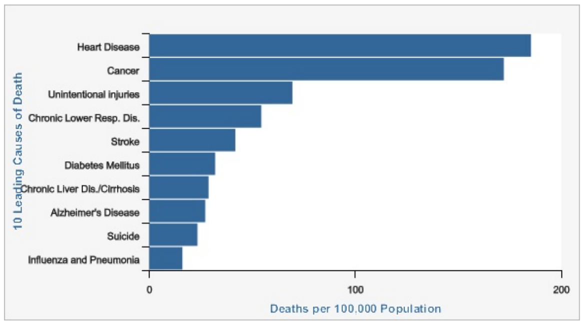


Figure 2- Ranked leading causes of death in New Mexico, 2017 [8]

Definition of Problem:

Diabetes is a widespread health problem that is a result of one's entire lifestyle. Our project aims to identify demographic factors that correlate with diabetes rate data in New Mexico counties and visualize their effect in as much detail as possible. Society as a whole needs to be aware of the demographics that affect diabetes rates on a macro level. It is on the macro level that a general population can be encouraged to live an anti-diabetic lifestyle. A visual model is an effective way of analyzing demographics as well as educating others on their impacts on diabetes. Our ideal result is a program that is capable of correctly predicting the diabetes rate of a town based on inputted factors.

Data Management:

To determine what variables have a correlation with diabetes rates, we found scatter plots to be the most effective technique. They are quick to analyze and simple to produce. In total, eight variables were analyzed. If the scatter plot showed a correlation, the variable was implemented into DANN (our artificial neural network). Graphs can be found in the Appendix. The diabetes csv data that is used is the 2017 diabetes rates for all thirty-three New Mexico counties from the CDC [16]. The other demographic data that was compared to the diabetes rates is from the U.S. Census Bureau [14]. The various figures that were collected from the Census range from 2017-2019.

Using Python to Analyze .csv Files

We found .csv to be a useful format for our data collection. Python has a csv module that can be easily used to extract data from a csv file, and the format is widely utilized across many sources like the U.S. Census and the CDC [2]. It was decided that using

Python is the best technique to create the scatter plots because a computational program is more capable of facilitating the data for all thirty-three New Mexico counties- a task that would be tedious using a method that requires manual entry. Eight programs were made to make the scatter plots, one for each variable. Each program works similarly- the only parameters that differ are the name of the .csv files, the name of the row that is being extracted, and the cosmetic details of the scatter plot such as the title. Matplotlib is a library that allows for comprehensive data visualization in Python. Once the programs have recorded the data from the csv files, they use Matplotlib to produce a scatter plot. After the scatter plot was made, it was manually determined if a correlation was present. **Scatter Plot Overview:** Considering the ambiguity of diabetes rates, we found several interesting correlations in the scatter plots. The variables that we concluded correlated with the diabetes rates in New Mexico are persons without health insurance, percent of American Indian and Alaska native alone, mean commute time, education (percent of population with high school or higher), and poverty rates. Note that the diabetes rate data includes both type 1 and type 2 diabetes. As stated in “Diabetes”, roughly 95 percent of diabetes diagnoses are type 2. Due to the unavailability of data that only contains the rate of type 2 diabetes, we must compare the variables to the whole diabetes rate. A situation where this could have interfered with our results could be in the education level scatter plot. Education would not affect type 1 diabetes, for no one knows how to prevent it. Since ethnicity has been found to play a role in both types of diabetes, the plot comparing diabetes to percent of American Indian and Alaska Native may be more viable than other tests that include a variable that doesn’t contribute to type 1 as well. The plot with American Indian and Alaska Native shows almost no correlation. It was surprising that there was not a more significant correlation for that variable since ethnicity plays a major role in both types of diabetes. American Indian/Alaska Native adults are nearly three times more likely than Caucasian adults to be diagnosed with diabetes [15]. It was concluded that because of the known connections between those ethnicities on diabetes, and due to numerous outliers that support a positive correlation, percent of American Indian and Alaska Native county data would be used in our neural network. Persons without health insurance under age 65 is an interesting demographic. Since diabetes is often the result of how someone lives their life, it was necessary to have a way of measuring how accessible medical services are to the counties' populations. We also wanted to potentially measure how much individuals in the area value their health involuntarily or deliberately with one's attitudes. Unexpectedly, there is an apparent positive correlation between the insurance data and diabetes rates.

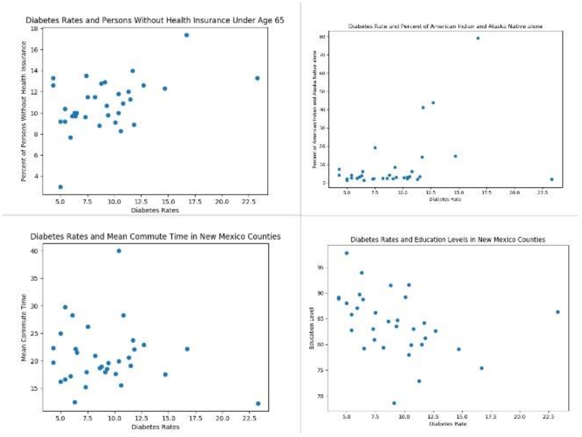


Figure 3- Four of the scatter plots that were used to test demographic variables with diabetes rates. Refer to the Appendix for more graphs

Computational Method:

About Mellitus:

Our goal when creating Mellitus was to create a simple, easy to interpret, model that is as close to a real community as possible. In many aspects, this has been achieved. There have been seven versions of Mellitus. Mellitus is a Latin word for Diabetes. It is built in Netlogo 6.1.1. Over each version, it has been gradually improved. Figure 4 shows the difference between the first version and the final version of Mellitus. There is a significant improvement in the usability, functionality, visual appearance, and overall accuracy of the model. Version 1.0 was based on little real-world data. Version 5.1 has proven to be quite useful in predicting a logical diabetes-rate, given the appropriate variable values. A central element to Mellitus is DANN (Diabetes Artificial Neural Network), which is imported with the py extension built into Netlogo. We needed a way to properly measure the number of people that are active in a community whether they are travelers or residents. For example, a town like Tucumcari, NM, has a relatively low number of residents but a high number of travelers due to its proximity to Interstate 40. This affects a community's demographic figures and can be an obstacle when

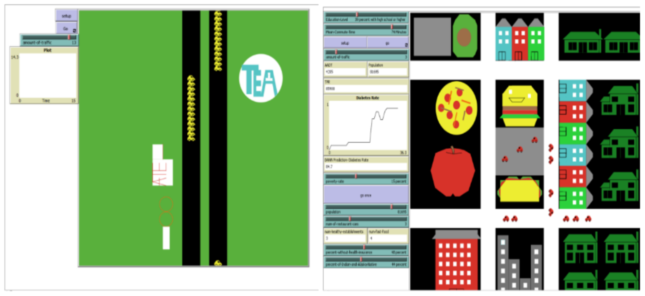


Figure 4- Comparison between Mellitus 1.0 and Mellitus 5.1

analyzing data. In order to appropriately compare demographics, we created a metric called TPR (total population rating). It is the area's annual average daily traffic (AADT) from the New Mexico Department of Transportation plus the community's population. **Mellitus Variables:** Factors that

Mellitus is capable of analyzing include education, commute-time, poverty, percent of population without health insurance, and percent of Indian and Alaska Native. These are all treated as independent variables, each one controlled by the user. The result is a change in the DANN diabetes rate prediction. The factors that were inputted into the neural network have been determined by producing and analyzing scatter plots. See "Scatter Plot Overview" for more information on this process. Broadband internet access and median household income did not have a viable correlation with diabetes rates in New Mexico. **Custom Turtles:** Custom Turtle shapes have been created to use instead of the default shapes in Netlogo. These are used to symbolize the demographics of the town. For example, when the TPR is raised more residential facilities appear in the interface.

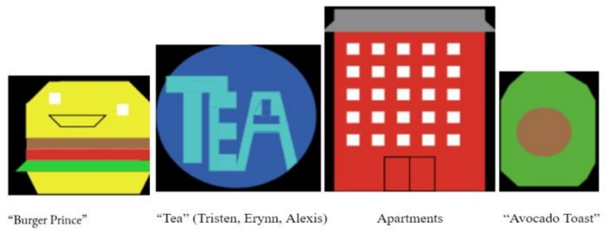


Figure 5- Several custom turtles that are used in Mellitus

Mellitus(5.0):

Mellitus 5.0 is the most polished version of Mellitus that doesn't utilize DANN. This means that the py extension is not required. Instead of DANN, Mellitus 5.0 uses linear equations derived from the lines of best fit of the scatter plots created. Scatter plots were also used to determine the functionality dynamics of Mellitus. Since our project aims for Mellitus to symbolize a real-world town as accurately as possible, we made scatter plots visualizing TPR and variables like number of fast-food restaurants or number of healthy-food restaurants using a similar method that was discussed in “Data Management”. Then lines of best fit for the scatter plots were drawn, and the linear equations were implemented. For example, the scatter plot comparing the number of fast-food restaurants in a town, and its TPR showed that one fast-food restaurant tends to serve about 5000 TPR. In Mellitus 5.0, every 5000 TPR, a fast-food restaurant appears in the town. The lines of best fit were converted from slope-intercept form to x-equals so that their outputs could be utilized in Mellitus. Mellitus 5.0 also uses this method for the independent variables that contribute to the diabetes rate like education, poverty, and commute time. Its diabetes rate prediction is not accurate to the real world, but that was not our intention in this version. It outputs a reasonable number that changes appropriately based on how the variables are changed by the user. We created it to have a version of Mellitus that doesn’t require the Python extension. It greatly improves the accessibility of our research and makes Netlogo Web or a computer without Python an option for anyone that would otherwise not be able to use Mellitus. That’s also why 5.0 has essentially the same interface as 5.1 and is capable of analyzing many of the same variables.

Mellitus(5.1):

The central result of our project is Mellitus 5.1. It has many similarities to Mellitus 5.0, but its main upgrade is support for DANN 2.0. The scatter plot derived linear equations still control the visual functionality of the model, like how many restaurants are open in the town. However, DANN brings the ability for Mellitus to compute far more powerfully with a non-linear approach to the diabetes rate. Refer to “DANN(2.0)- Sigmoid” for an in-depth explanation on DANN 2.0; this section will focus on its execution in Mellitus 5.1. The program takes the value of each variable from the sliders and puts them in a format that can be read by DANN using the py:set command. DANN 2.0 is imported into Mellitus by using the py:run command. The diabetes rate prediction is made using py:runresult. Py:set is capable of giving a variable in a Python script a Netlogo value. Py:runresult converts Python values to Netlogo values. In Mellitus, it is used to store the float output of DANN in a Netlogo variable. Py:run is able to run the Python script required for the program. It allows a Python script to be used in Netlogo. DANN runs continuously in Mellitus, and its output is displayed in the interface on a monitor and a plot. When the sliders are adjusted the data is easily analyzed by the user, and a correlation can be made with the plot. Refer to the “Evaluation and Testing” section of “DANN(2.0)- Sigmoid” for an overview of the performance of DANN 2.0. Mellitus proves to be an effective medium for exhibiting the output of DANN 2.0, and shows far more and more detailed information than would be possible in native Python. The plot tool in Netlogo allows correlations between the slider controlled independent variables and changes in the virtual diabetes rate to be clearly made. Since DANN is constantly running in Mellitus, the virtual diabetes rate is constantly adapting and factors visually link before the user’s eye. We couldn’t have achieved this using any other method. In many situations, the correlations made by Mellitus are outstanding and even somewhat contradict the correlations made by our scatter plots. The scatter plot with poverty data suggests a positive correlation between county poverty rates and diabetes rates, while generally, Mellitus suggests a negative correlation between the two. Remember that Mellitus does not function linearly and does not always behave this way. Previously, we mentioned that it was surprising to see such an insubstantial correlation between the percent of American Indian and Alaska Native county data and diabetes rates. This variable is widely known to affect one's odds of being diagnosed with diabetes, so we were not surprised when Mellitus showed that it had a positive correlation with diabetes rates. However, even in Mellitus, it does not appear to affect the diabetes rate dramatically. The other variables mostly perform in the ways that were suggested by the scatter plots when operating in Mellitus. For instance, Mellitus exhibits a strong positive correlation between education levels and diabetes rates.

Neural Network:

We had originally planned to use a Chi-Square Test to analyze the variables and their effect on diabetes rates, however a few problems occurred. A Chi-Squared Test is routinely used to test if a distribution of categorical variables is independent. This approach won't work because we aren't analyzing categorical data, we're analyzing a numerical disease rate. A Neural Network is a useful alternative because diabetes rates are determined by the lifestyle of a

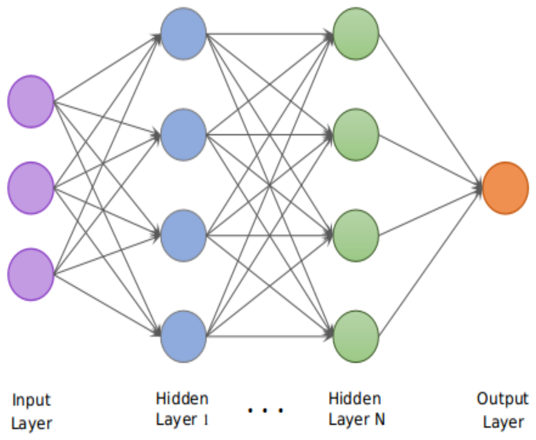


Figure 6- Diagram of artificial neural network

population, a wide range of factors to study. It could potentially observe enough frequency in the data to make a viable prediction off it. A Neural Network could also be implemented into Mellitus fairly easily with the Netlogo py extension. Our Neural Network is named DANN (Diabetes Artificial Neural Network

About Neural Networks:

Artificial Neural Networks are ambiguously based on the biological networks that compose a human brain. They can solve problems less like a computer and more like a human, without the need for explicit commands [4]. Instead, they analyze example data to learn. This process is especially useful when evaluating abstract figures like factors that contribute to diabetes rates. A Neural Network is made up of 3 main layers; an input layer, *n* number of hidden layers, and an output layer [18]. If there is more than one hidden layer, it is considered a Deep Neural Network. When values are transmitted through neurons, the weights are applied to the values and passed into the activation function with the bias. It is widespread for weights and biases to be set to random values initially so that they can be adjusted later on by the program. Parameters that cannot be regulated by the program itself are called hyperparameters. This includes learning rate, activation function, loss function, and more. The learning rate controls how quickly the network is acclimated to a data pattern [1]. If the learning rate is too high, the output may not be as accurate as anticipated. However, if it’s too low, the program could crash during the training process. The activation function is very important. Some of the most popular activation functions include Sigmoid, RELU, Tanh, and Softmax, usually in the output layer and often when classifying objects. Activation functions calculate a weighted sum and add bias to it. They're the reason for Neural Networks' capability of abstract thinking [6]. They're so important that there is a dedicated section in this paper for them. The loss function optimizes the parameters in a Neural Network [10]. It compares the "correct" value from training data with the value that the network predicted so that backpropagation can occur, and parameters can be updated. More Hyperparameters that are important to this project include the optimizer, number of epochs, and batch size. Optimizers refresh the weight parameters to minimize the loss function. The loss function works by advising the optimizer on how accurate it is [17]. Epochs are the number of times the neural network passes through all the training data. Note that there is a significant difference between iteration and epoch. Iteration is the number of passes (one pass = one forward pass + one backward pass), an epoch is the number of times the entire training data has been viewed by the program. Batch size is the number of data samples that are propagated through the network in one iteration. **Activation Function:** The choice of the activation function is vital to receive desirable results, and the right choice is dependent on the problem being solved. For example, the Softmax Function is practical in situations where classification is required. We determined that the RELU (rectified linear unit) function is best for our project because it is capable of working with numbers higher than one, unlike the Sigmoid Function. The Sigmoid Function is useful for percentile data which can be converted to decimals between zero and one. It is what is used in DANN 2.0. The Sigmoid Function can also be built manually in Python, without the need for external

APIs like Keras. This also means that execution in Netlogo is possible. Due to a bug in the Netlogo py extension, DANN 3.1, which uses the RELU function, cannot be applied to Mellitus. See “DANN(3.1)-RELU” for information on DANN 3.1. The linear activation function is simple yet essential to our project. It is essentially a linear regression model ($y = ax$). It wouldn’t be beneficial to use in the hidden layers because it isn’t capable of advanced calculations. However, it is useful in the output layer. **About Tensorflow and Keras:** Tensorflow is an open-source software library developed by Google for machine learning execution, including neural networks. Keras is also an open-source library. Its primary purpose is to create neural networks. Both APIs were used in Python. Keras works on top of Tensorflow [5]. Keras has been implemented in DANN 3.1 so that the RELU activation function can be used. Using Keras is relatively simple. We made a Sequential model, which is a linear stack of layers. Packages that were implemented in our project include: add, compile, fit, and predict. Add is used for adding new layers into the sequential model. Compile configures the model for training. Fit trains the model with a number of epochs. Predict is used to return predictions from the model.

DANN(2.0) – Sigmoid:

DANN 2.0 is the first stable edition of DANN and was built in Python 3.7. It uses Sigmoid as the Activation Function. This decision brings advantages and disadvantages. It has a domain of all real numbers and a range of 0 to 1. As long as the input values are proportional, it is capable of monotonically adapting the values so that they are between 0 and 1 and can be used by the network. This is practical for comparing percent variables. All significant variables were percentage values or had a form of percentage alternative. For example, percent of persons in poverty, percent of persons with a high school education or higher, and percent of persons without health insurance are all variables that contribute notably to diabetes. No external machine learning modules were used in this version. However, Numpy was used: a package for scientific computing with Python [7]. The Sigmoid Function was manually utilized using NumPy by inserting the Sigmoid equation into the program. One of the packages that was used is Exp, which is capable of calculating the exponential of elements in an array [9]. It is established on Euler’s number, an irrational number that is the base for the natural logarithm. Another is Array, which is useful for data handling and putting values into arrays. Also, Dot is a very important tool, the equivalent to matrix multiplication. The data was inputted in

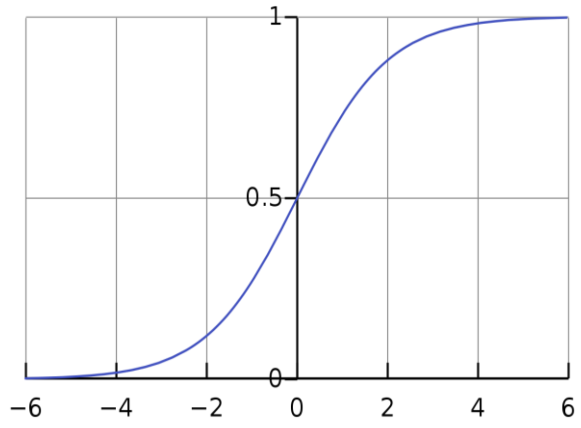


Figure 7- Graph of Sigmoid function

arrays. Twenty counties were used as training data. There are thirty-three counties in New Mexico. We determined that twenty training counties are most desirable so that we could retain a large and diverse set of testing data. DANN 2.0 analyzes education, poverty, persons without health insurance, percent American Indian and Alaska Native, and commute time divided by 100. The commute time data is divided by 100 for the Sigmoid function. Since the commute time dataset used to train the network is also divided by 100, this should not interfere with the accuracy of its prediction. Implementing the percent of American Indian and Alaska Native into DANN should also help eliminate the potential interference of type 1 diabetes. As stated before, there is nothing that is known to prevent type 1 diabetes, and about 95 percent of diabetes cases are type 2. Since a factor that contributes to both types of diabetes is ethnicity and the data that was used to train the network is the rate of both types of diabetes, the American Indian and Alaska Native variable may contribute extensively toward DANN's output. DANN 2.0 is especially minimalistic. One neuron is modeled with five input connections and one output connection. When operating in a native Python IDE, we set the iterations at 300,000. This is more than enough for the program to come to a reliable conclusion. However, we have observed a noticeable decline in output accuracy when lowering the iterations. In Mellitus, the iterations are set to 100,000 to minimize the amount of lag. The reason why this version of DANN plays an important role in our project is that it is the only version of DANN that is functional in Mellitus. (see "Netlogo Execution Shell Bug" for more information about why other versions of DANN cannot be used in Mellitus). **Evaluation and Testing:** The performance of DANN 2.0 varies. In some situations, it is capable of predicting the diabetes rate almost exactly, while in others, it fails to reach this goal. Note that Table 1 shows the results from one trail, inputting the appropriate data for each county. We chose Los Alamos and De Baca county to test with because each has relatively unique statistics. For instance, Los

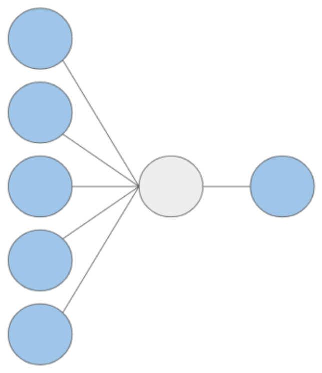
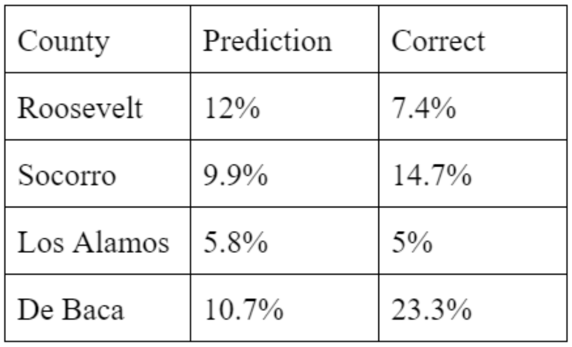


Figure 8- Diagram of DANN 2.0

Alamos County has a distinctively high education level, while De Baca county has the highest diabetes rate in New Mexico. The most prominent figure in Table 1 is the predicted diabetes rate of Los Alamos County. It was essentially able to predict the diabetes rate with perfect accuracy. This may be due to an anomaly in the singular trial. We did use “seed()” in the Python random module so that the random number output would remain the same. As stated in “About Neural Networks”, weights and biases are initially set to random values and adjusted later by the network itself. Using



County	Prediction	Correct
Roosevelt	12%	7.4%
Socorro	9.9%	14.7%
Los Alamos	5.8%	5%
De Baca	10.7%	23.3%

Table 1- Accuracy testing of DANN 2.0

Seed means that the program begins with the same weights and biases every time it runs and therefore outputs a similar value every run given comparable data. If Seed was removed, we could see more or less accurate predictions, but they would certainly be far less consistent.

DANN(3.1) -RELU:

DANN 3.1 is the most advanced and complex version of DANN. It utilizes the RELU activation function, allowing for whole number data output. Keras is implemented and introduces access to many Hyperparameters that are not available in DANN 2.0. This allows us to fine-tune the neural network to produce the best results. In contrast to DANN 2.0, this version is also a Deep Neural Network, with two hidden layers. It works with the same dataset as 2.0 with twenty counties and five variables.

However, it is capable of keeping the values in their original format instead of requiring them to be divided by 100. There are five nodes in each hidden layer. The RELU function is used in all layers except in the output layer. The output layer makes use of a classic linear activation function. After observing the performance of the network in several circumstances, we found that it was most accurate when the output layer lacks the RELU function. This means that the main calculations are taking place in the input and hidden layers, and the output layer reflects the result of these calculations with a linear regression. The program has 45 epochs and a batch size of 5. There are 20 training examples, so it takes four iterations to complete one epoch. The program has 180 iterations in total. This number is far less than in 2.0, but 3.0 is able to complete its iterations more effectively due to many improvements, like additional adjustable hyperparameters. Its loss function is a commonly used regression loss function, mean-squared-error (MSE), also known as mean-squared-deviation (MSD). The

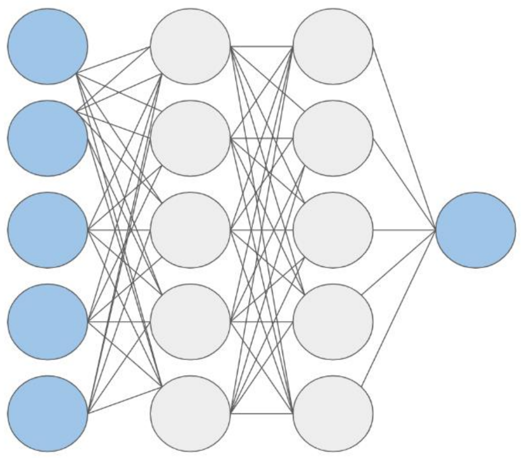


Figure 9- Diagram of DANN 3.1

program's optimizer is the Stochastic Gradient Descent (SGD) optimizer, which brings support for new hyperparameters like momentum and learning rate. Since there is no exact method to determine where the hyperparameters should ideally be set, various frameworks were experimented with, and it was eventually concluded that the most desirable results are achieved with the learning rate set to 0.001 and the momentum set to 0.7. DANN 3.1 is a Sequential Keras model. Refer to "About

Tensorflow and Keras” for explanations of the Keras packages that were used to construct the network. Like 2.0, it was built in Python 3.7. **Evaluation and Testing:** DANN 3.1 has proven to be considerably more accurate than DANN 2.0. In this version, the program tends to output a different prediction every run for the counties. This is due to the program adjusting parameters differently each run. No tool is utilized to use the same random values every run like Seed was used in DANN 2.0. Although this version contains many more hyperparameters, weights and biases still start off with random values. DANN 3.1 was more accurate than DANN 2.0 in most counties in which it was tested. This is likely due to all of the improvements that were discussed, like an additional hidden layer with more neurons and the RELU activation function. Table 2 shows the behavior of DANN 3.1 from one trial.

County	Prediction	Correct
Roosevelt	7.8%	7.4%
Socorro	15.9%	14.7%
Los Alamos	6.8%	5%
De Baca	16.4%	23.3%

Table 2- Accuracy testing of DANN 3.1

Conclusion:

Results:

Mellitus is a practical template for accurately modeling a town, its demographics, and its diabetes rate. Factors have been identified that contribute to diabetes and have been manifested in a computational method. Mellitus is especially useful for analyzing the output of DANN 2.0. When using a traditional Python interpreter to run DANN, the diabetes rate prediction is outputted a single time per run.

In Mellitus, DANN can be used, not only to predict a diabetes rate but to identify which and to what significance input variables are contributing to the prediction. It can be concluded from the many techniques that we used that the variables that have a significant impact on diabetes rates in New Mexico counties are, from least significant to most: percent of American Indian and Alaska Native, poverty, education, commute time, and health insurance. This was determined using the plot in Mellitus 5.1, visualizing the virtual diabetes rate. Each variable's slider was adjusted to a value that resembled a proportional half-way value. For instance, if the variable was a percentage, it was adjusted to fifty percent. If the variable was a numerical value, the slider was adjusted to half-way. This process was applied to one variable at a time, while the other variables were set to zero. Of course, this isn't an effective way of simulating a real-world town, but it is sufficient for determining how much each variable's impact is on the diabetes rate. Then it was possible to visualize a change in the virtual diabetes rate using the plot. A more significant change in

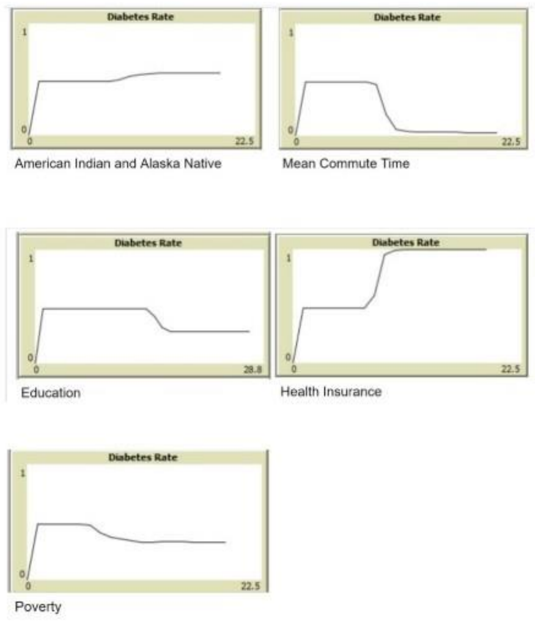


Figure 10- Mellitus 5.1 DANN output visualization

the slope of the line indicates a more significant effect on diabetes rates. Figure 10 shows the results of this experiment. A surprising aspect of this test is the negative correlation between mean commute time and diabetes rates. The previously shown scatter plot suggested a positive correlation. It is unclear why DANN reacted to the scenario this way, but it could be due to a shortage of training data or the lack of the

RELU activation function. As stated earlier, the commute time value in DANN 2.0 has to be divided by 100 so that it works with the Sigmoid function. However, since the commute time training data is also divided by 100, this should not interfere with our results.

Netlogo Execution Shell Bug:

Due to a bug in Netlogo 6.1.1, Mellitus is unable to support DANN 3.1. In contrast to 2.0, 3.1 requires external packages like Keras and Tensorflow. Anaconda is a popular data science platform that contains all of the packages that Mellitus requires, so we decided to use an Anaconda virtual environment with the Netlogo py extension. When using an Anaconda virtual environment, the program gives an error message that states "Extension exception: This is a bug. Please report". The appropriate path to the environment was entered in the Python tab in Netlogo. If the path to the Python executable is empty, Netlogo will attempt to find the appropriate executable. When the executable is empty, Netlogo will show various bug messages, and the packages that DANN 3.1 requires still cannot be used. In an attempt to resolve this issue, Mellitus was translated to several versions of Netlogo other than 6.1.1 to detect if the program reacts differently. In every version, the program showed an error message, each stating something along the lines of a bug existing in Netlogo.

Suggestions for Further Implementation:

In the future, more variables should be implemented into Mellitus. DANN can be enabled to output an even more accurate diabetes rate by expanding the data set to include all 33 New Mexico counties. Currently, DANN evaluates 20 counties, which is not as desirable as one would like. This concept could even be applied outside New Mexico. Web scraping could be used to collect mass amounts of data on subjects such as fast-food or recreational facilities and can be used on a global scale. We would also like to conquer the execution shell bug in Netlogo and have DANN 3.1 support in Mellitus. If we could gain access to a large database that includes the data that is needed on a national or global scale, DANN could be trained to perfection and may even have mainstream usage in biomedical and government administrative fields. If Mellitus is correct, ethnicity has less effect on diabetes than other factors that can be controlled. When changes are being made that alter demographic variables in a town, people can be educated of the domino effect that comes with them. Many factors that benefit the general New Mexican society also help the cause for lower diabetes rates. This includes more access to education and affordable health insurance. The first step has already been taken in expanding our project by making Texas versions of DANN. See the Appendix for links and information.

Acknowledgements:

We would like to thank Mr. Alan Daugherty from Melrose for introducing us and guiding us through the Supercomputing Challenge. We would like to thank Mrs. Leah Lee for being our sponsor teacher. We would also like to thank our parents for showing support throughout this project. Also, we would like to thank Mrs. Elnabeth Grau, our school Superintendent, and Mrs. Michell Edwards, our principal, for all their support. Diane Lauderdale from the University of Chicago Biological Sciences contributed significantly to our project by counseling us against the use of the Chi-Square Test, which we now know is ineffective for our uses. Instead, we created DANN and took a machine learning approach to our problem. Thank you to Mrs. Jennie Piepkorn for proofreading the Final Report and assisting us in revision.

References:

[1]. Brownlee, J. (2020, February 5). Understand the Impact of Learning Rate on Neural Network Performance. Retrieved from https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

[2]. Data & Statistics. (2019, May 30). Retrieved from https://www.cdc.gov/diabetes/data/index.html

[3]. Diabetes. (n.d.). Retrieved from https://www.who.int/news-room/fact-sheets/detail/diabetes

[4]. Frankenfield, J. (2020, January 29). What Are Artificial Neural Networks? Retrieved from https://www.investopedia.com/terms/a/artificial-neural-networks-ann.asp

[5]. Keras: The Python Deep Learning library. (n.d.). Retrieved from https://keras.io/

[6]. Khandelwal, R. (2019, February 4). Overview of different Optimizers for neural networks. Retrieved from https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3

[7]. NumPy. (n.d.). Retrieved from https://numpy.org/

[8]. Health, D. of. (2017). Complete Health Indicator Report of Death Rates from Leading Causes of Death. Retrieved from https://ibis.health.state.nm.us/indicator/complete_profile/DthRateLdgCause.html

[9]. numpy.exp¶. (n.d.). Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.exp.html

[10]. Sakshi, T. (2018, February 6). Activation Functions in Neural Networks. Retrieved from https://www.geeksforgeeks.org/activation-functions-neural-networks/

[11]. Sullivan, D. (2018, October 25). Cost of Type 2 Diabetes. Retrieved March 19, 2020, from https://www.healthline.com/health/cost-of-diabetes#1

[12]. The Burden of Diabetes in New Mexico. (n.d.). Retrieved March 18, 2020, from http://main.diabetes.org/dorg/PDFs/Advocacy/burden-of-diabetes/new-mexico.pdf

[13]. Type 1 Diabetes. (2020, March 11). Retrieved from https://www.cdc.gov/diabetes/basics/type1.html

[14]. U.S. Census Bureau QuickFacts: New Mexico. (n.d.). Retrieved from https://www.census.gov/quickfacts/NM

[15]. U.S. Department of Health and Human Services Office of Minority Health. (2017). Retrieved from https://minorityhealth.hhs.gov/omh/browse.aspx?lvl=4&lvlid=33

[16]. U.S. Diabetes Surveillance System. (n.d.). Retrieved from https://gis.cdc.gov/grasp/diabetes/DiabetesAtlas.html#

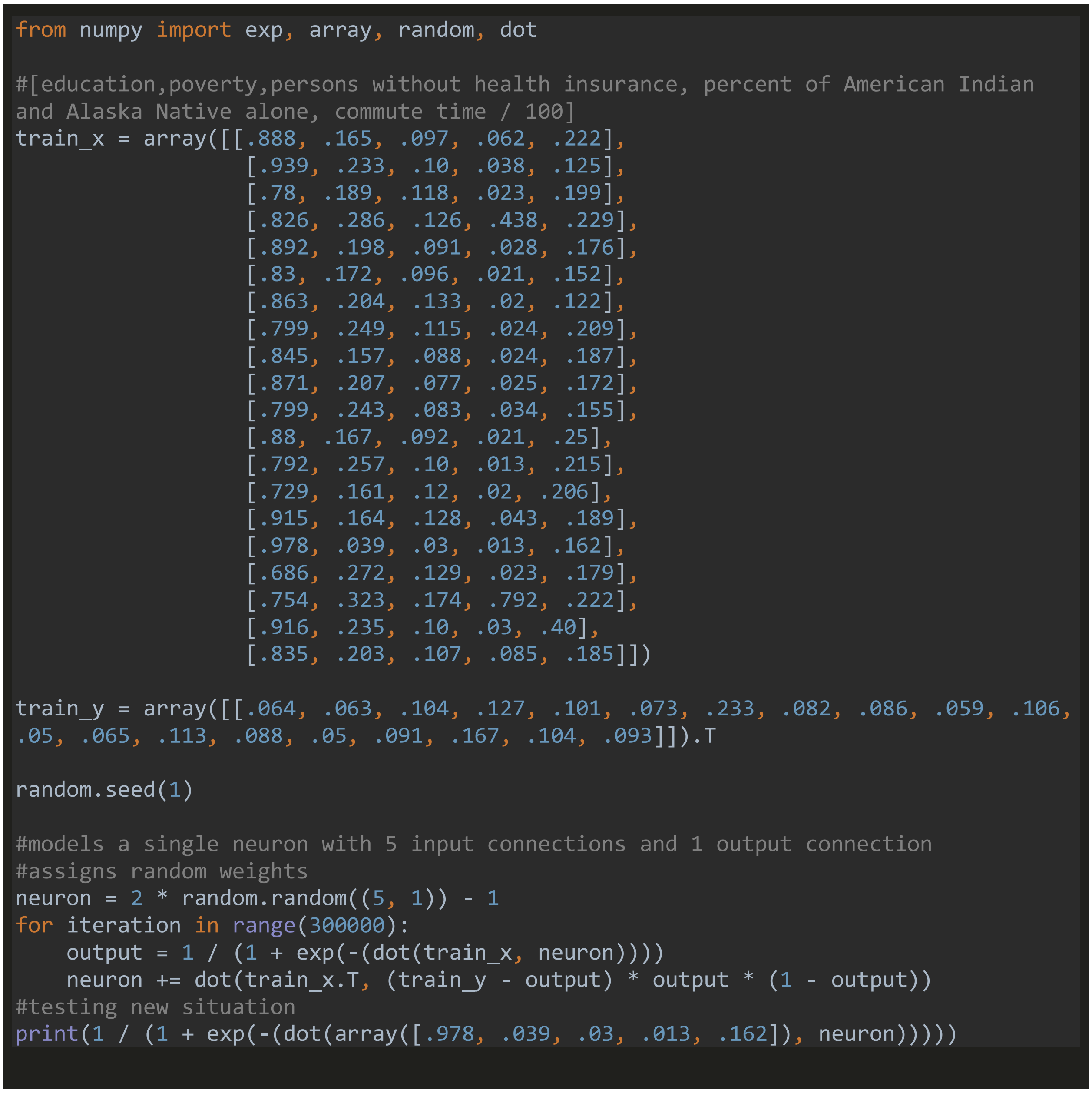
[17]. Verma, S. (2020, February 13). Understanding different Loss Functions for Neural Networks. Retrieved from https://towardsdatascience.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718

[18]. Weights & Biases - Fundamentals of Neural Networks. (n.d.). Retrieved from https://www.wandb.com/articles/fundamentals-of-neural-networks

[19]. What is Type 1 Diabetes? (n.d.). Retrieved from https://www.endocrineweb.com/conditions/type-1-diabetes/type-1-diabetes

Appendix:

DANN(2.0) Code:



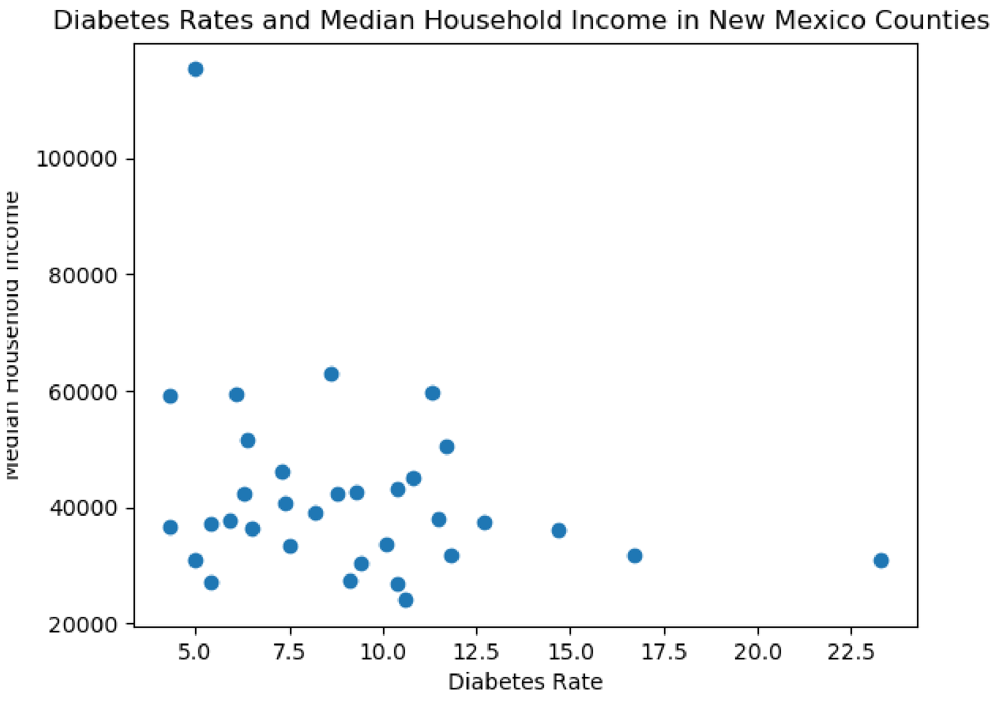
```
from numpy import exp, array, random, dot

#[education,poverty,persons without health insurance, percent of American Indian
and Alaska Native alone, commute time / 100]
train_x = array([[.888, .165, .097, .062, .222],
                  [.939, .233, .10, .038, .125],
                  [.78, .189, .118, .023, .199],
                  [.826, .286, .126, .438, .229],
                  [.892, .198, .091, .028, .176],
                  [.83, .172, .096, .021, .152],
                  [.863, .204, .133, .02, .122],
                  [.799, .249, .115, .024, .209],
                  [.845, .157, .088, .024, .187],
                  [.871, .207, .077, .025, .172],
                  [.799, .243, .083, .034, .155],
                  [.88, .167, .092, .021, .25],
                  [.792, .257, .10, .013, .215],
                  [.729, .161, .12, .02, .206],
                  [.915, .164, .128, .043, .189],
                  [.978, .039, .03, .013, .162],
                  [.686, .272, .129, .023, .179],
                  [.754, .323, .174, .792, .222],
                  [.916, .235, .10, .03, .40],
                  [.835, .203, .107, .085, .185]])

train_y = array([[.064, .063, .104, .127, .101, .073, .233, .082, .086, .059, .106,
.05, .065, .113, .088, .05, .091, .167, .104, .093]]).T

random.seed(1)

#models a single neuron with 5 input connections and 1 output connection
#assigns random weights
neuron = 2 * random.random((5, 1)) - 1
for iteration in range(300000):
    output = 1 / (1 + exp(-(dot(train_x, neuron))))
    neuron += dot(train_x.T, (train_y - output) * output * (1 - output))
#testing new situation
print(1 / (1 + exp(-(dot(array([.978, .039, .03, .013, .162]), neuron)))))
```

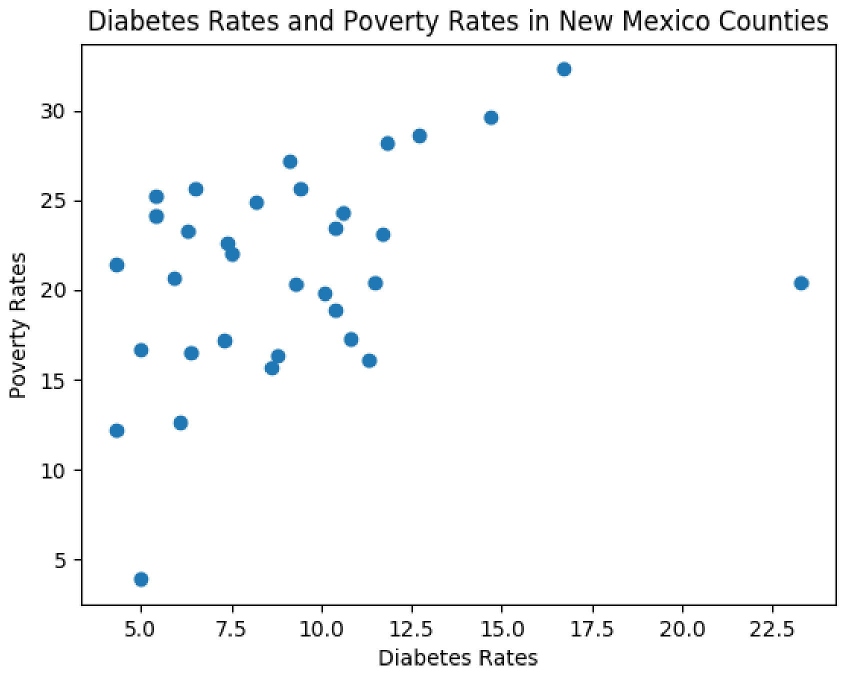
DANN(3.1) Code:

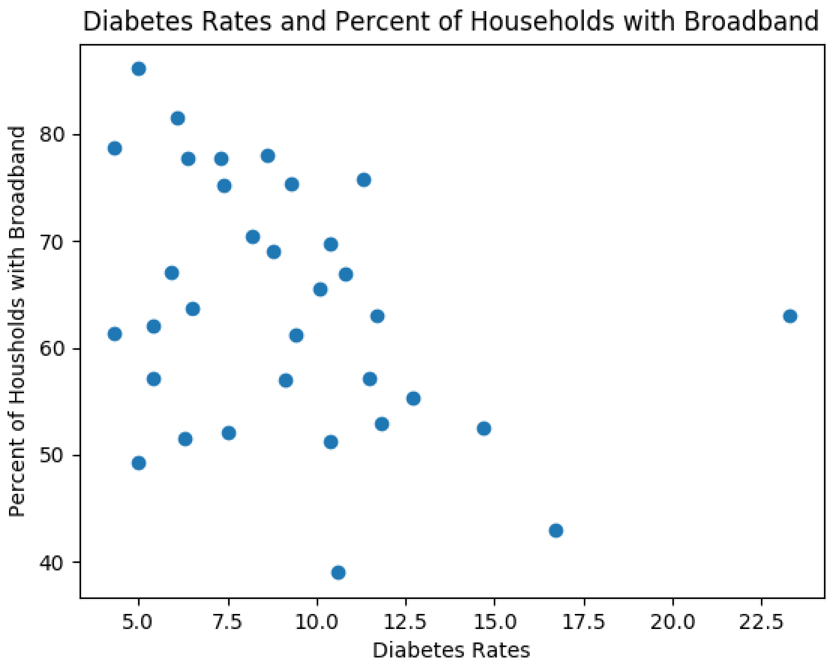
```
import keras
from keras.models import Sequential
from keras.layers import Dense
from numpy import array

#[education,poverty,persons without health insurance, percent of American Indian
and Alaska Native alone, commute time]
x_train = array([[.888, .165, .097, .062, 22.2],
                [.939, .233, .10, .038, 12.5],
                [.78, .189, .118, .023, 19.9],
                [.826, .286, .126, .438, 22.9],
                [.892, .198, .091, .028, 17.6],
                [.83, .172, .096, .021, 15.2],
                [.863, .204, .133, .02, 12.2],
                [.799, .249, .115, .024, 20.9],
                [.845, .157, .088, .024, 18.7],
                [.871, .207, .077, .025, 17.2],
                [.799, .243, .083, .034, 15.5],
                [.88, .167, .092, .021, 25],
                [.792, .257, .10, .013, 21.5],
                [.729, .161, .12, .02, 20.6],
                [.915, .164, .128, .043, 18.9],
                [.978, .039, .03, .013, 16.2],
                [.686, .272, .129, .023, 17.9],
                [.754, .323, .174, .792, 22.2],
                [.916, .235, .10, .03, 40],
                [.835, .203, .107, .085, 18.5]])

y_train = array([[.064], [.063], [.104], [.127], [.101], [.073], [.233], [.082],
[.086], [.059], [.106], [.05], [.065], [.113],
[.088], [.05], [.091], [.167], [.104], [.093]])

x_test = array([[.891, .214, .126, .075, 19.7]])

model = Sequential()

model.add(Dense(units=5, activation='relu', input_dim=5))
model.add(Dense(units=5, activation='relu'))
model.add(Dense(units=1, activation="linear"))

model.compile(loss='mean_squared_error', optimizer=keras.optimizers.SGD(lr=0.001,
momentum=0.7, nesterov=True))
```

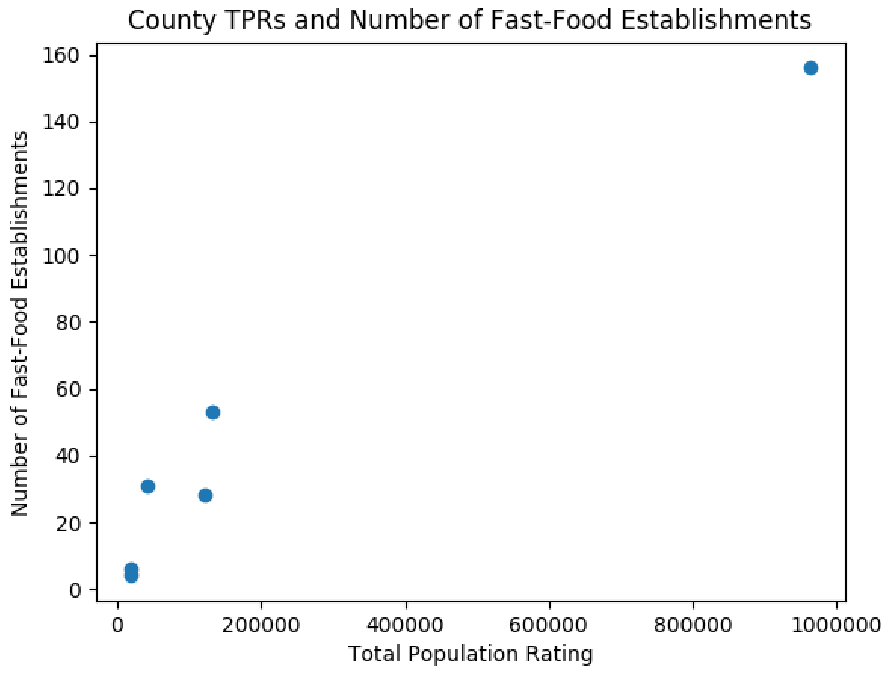
```
model.fit(x_train, y_train, epochs=45, batch_size=5)

print(model.predict(x_test, batch_size=1))
```

Additional Scatter Plots:




Diabetes Rates and Poverty Rates in New Mexico Counties
Poverty Rates
Diabetes Rates
30
25
20
15
10
5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5


Diabetes Rates and Percent of Households with Broadband
Percent of Housholds with Broadband
Diabetes Rates
80
70
60
50
40
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5



Link to Texas version of DANN- Sigmoid (Beta):
https://drive.google.com/open?id=1eHs_VlIwHTuSCZPXtLzBOyHbNiS_fblj

Link to Texas version of DANN- RELU (Beta):
https://drive.google.com/open?id=1gBCsa0TPQcAAjwsp-QPM8wyeZILA1Imu