Using an SVM and Neural Network to Predict Diabetes in Pima Native Americans

Samuel Silva
April 2020

1. Introduction

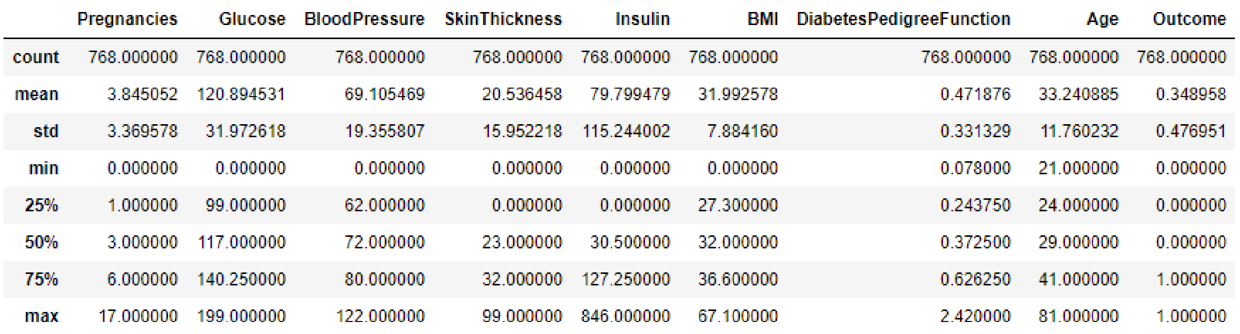
The data used in the models are originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The goal of the dataset was to predict if a patient had diabetes based on a variety of variables. A larger dataset exists but constraints were created that allowed for smaller datasets. The portion of the data used in this paper is restricted to patients being female of at least 21 years old and Pima Indian heritage [2].

2. Interpreting the Data

The data is made up of 9 columns, named: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome. The features that are eligible for machine learning models are all variables except Outcome. I want to be able to predict the Outcome using the remaining 8 features.

- **Pregnancies** refers to the number of times the subject is pregnant.
- **Glucose** is the measure of one's plasma glucose concentration after 2 hours in an oral glucose tolerance test.
- **BloodPressure** is the diastolic blood pressure in mm Hg
- **SkinThickness** is the measure of the thickness of triceps skin fold in millimeters
- **Insulin** measure after a 2-Hour serum insulin measured in mu U/ml
- **BMI** is body-mass index in (weight in kg/(height in m)^2)
- **DiabetesPedigreeFunction** is a function which scores likelihood of diabetes based on family history
- **Age** age in years

Outcome is the class variable (0 or 1) where 268 of 768 are 1, the rest are 0. A 1 suggests that the person has diabetes, a 0 suggests they do not have diabetes. Since I have classification data I can choose between multiple models. These models include logistic regression, nearest neighbors, supper vector machines (SVM), and neural networks. Not every feature is necessarily going to be helpful in the models, so it is important to look at correlations and descriptive statistics to choose the right features. It is also important to look at the ranges of the data. Typically, it is helpful to scale data when there are a lot of features or when the features are on very different scales. Looking at a summary of the data, I can use the ranges as a basis to determine if the data needs to be scaled.



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Just by looking at the means, standard deviation, and min and max, I can tell that the data has a mix of values, ranges, and dispersion, so it would be smart to scale the data for the models. Additionally, the minimum for many columns is 0 which seems improbable. Someone cannot have a blood pressure of 0 (unless they are dead), 0 skin thickness, 0 glucose, etc.. This means that I would have to do some cleaning of the data and remove false zeros. A scatter plot can also reveal false zeros as seen in Figure 1.

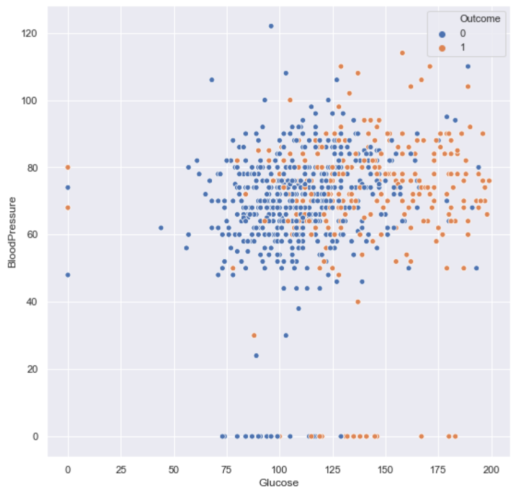


Figure 1: BloodPressure vs. Glucose: We can see that there are multiple data points suggesting that patients have a blood pressure of 0. This is fairly impossible, therefore the data has false zeros.

Since the data is binary, I plan on doing a SVM and a neural network. To choose what features I want to use in my SVM model, I can look at the correlation plot in Figure 2.

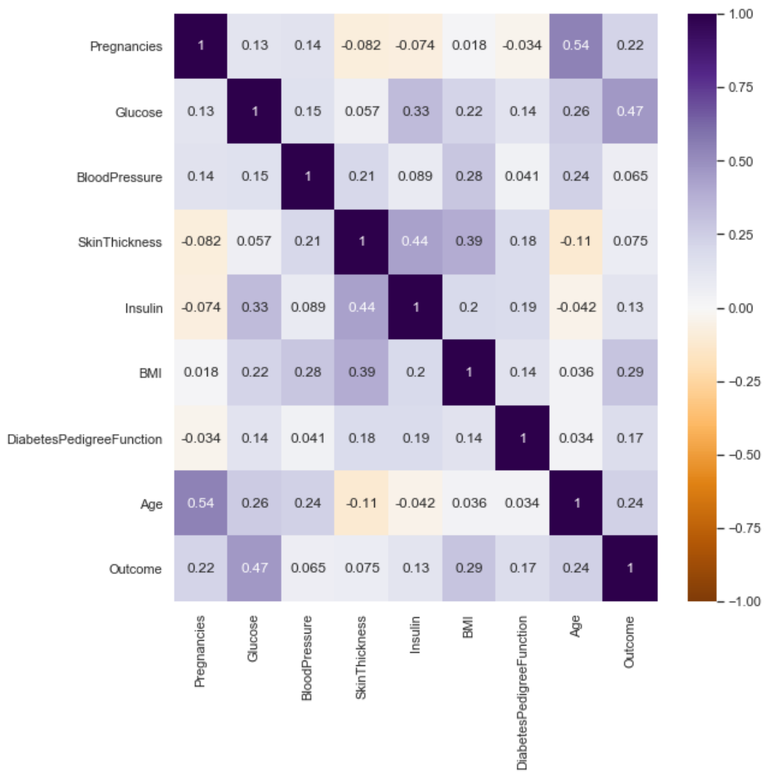


Figure 2: Correlation Map between all features in the data. BMI and Glucose have the highest correlations with Outcome.

Since diabetes is partially determined by one's blood sugar level, it would be a good idea to use **Glucose** as a feature. Furthermore, obese people are prone to diabetes so using the body mass index as a predictor for diabetes would be useful. Both **Glucose** and **BMI** have the highest correlations with **Outcome** so I will choose these features for my SVM model.

3. Creating a SVM Model

Similar to a logistic model, I can create a support vector machine (SVM) where the goal of the SVM algorithm is to find a hyperplane in an N-dimensional space that classifies the data points. There are countless hyperplanes that could separate the two groups of data. The goal is to find a plane that has the maximum margin between data points of both classes. This can be visualized in the example in Figure 3. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence, thus the model is more generalized.

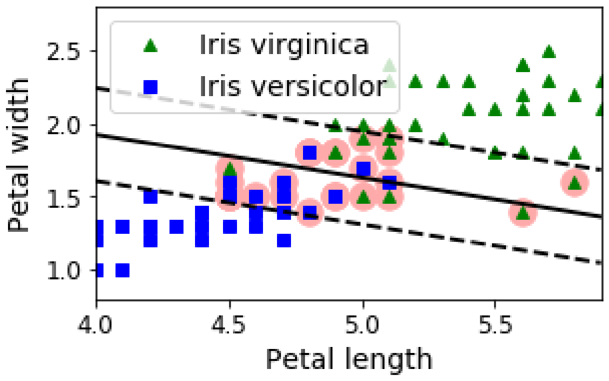


Figure 3: Sample decision boundary using an SVM model to maximize the margins between the data points and boundary allowing for greater generalization to new data.

Using **Glucose** and **BMI** as my features, I can begin constructing a SVM and eventually train it using my training dataset. There are three kernels that are used in SVM models, this includes linear, rbf, and polynomial. The type of kernel used is what effects the decision boundary. For example, Figure 3 has a linear decision boundary from the model's utilization of a linear kernel. An rbf kernel uses the radial basis function to construct the boundary between the data classes. A less common kernel is the polynomial which fits a polynomial function as a decision boundary.

The best model can be found using functions that try a variety of combinations and then measuring which parameters give the best accuracy. The model is fitted on training data during this time. Parameters that I can vary include the kernel, C, and gamma. The gamma parameter "defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'." C parameter trades off correct classification of training examples against maximization of the decision function's margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy. Essentially, C behaves as a regularization parameter in the SVM. I allowed the model to search over a variety of C's and gamma's using a logistic scale.

My most accurate SVM was a linear kernel, a gamma value of 0.0001 implying the data will vary more smoothly with landmarks, and a C value of 0.01 suggesting higher regularization leading to possible underfitting. Interesting enough, I received the same the same accuracy using an rbf kernel with C = 1000, and gamma = 0.01. A high C suggests low regularization. The models are seen in Figure 4. I received an accuracy of 0.754 on training data, 0.802 for testing. This low accuracy is not too surprising because we see overlap between the two outcomes, just by looking at the data it would be impossible to get a decision boundary that extremely accurately separated the two group.

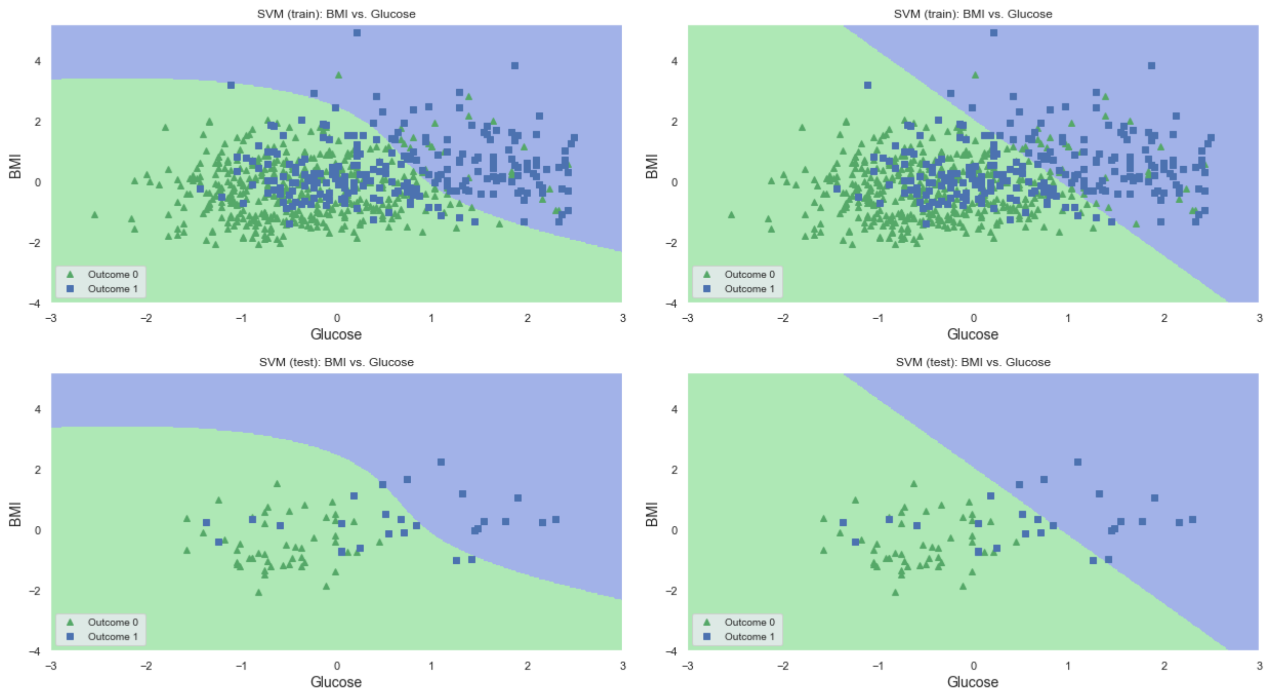


Figure 4: Results from SVM model: RBF kernel over training data (top left), Linear kernel over training data (top right), RBF over testing data (bottom left), Linear over testing data (bottom right)

Since both models gave the same accuracy on testing and accuracy, which model should be chosen? Neither model looks like it has overfitting or underfitting, so it seems reasonable to use either. However, I would choose the linear model because it predicts diabetes at higher glucose levels Figure 4. There is a chance that someone can have a high BMI but not have diabetes, but if you have an abnormally high glucose levels, then you will be diagnosed with diabetes; this is why I will choose the linear SVM model. Using the linear SVM, I can also look at a ROC curve to evaluate the quality of the model, Figure 5.

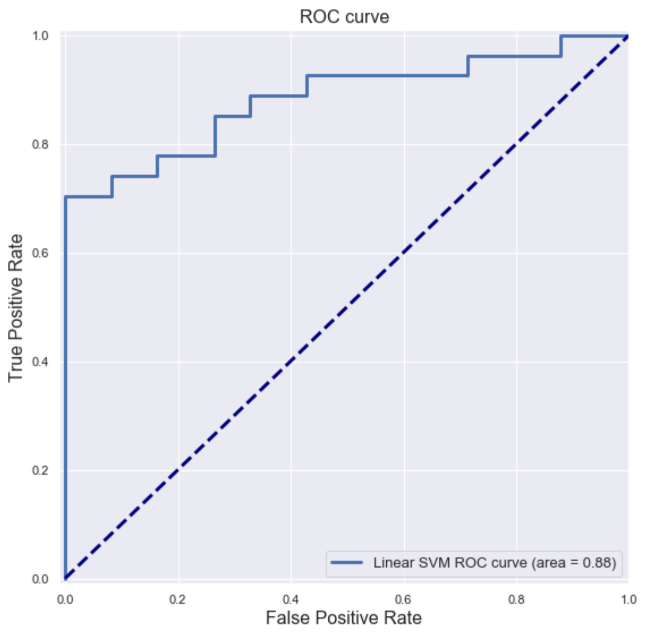
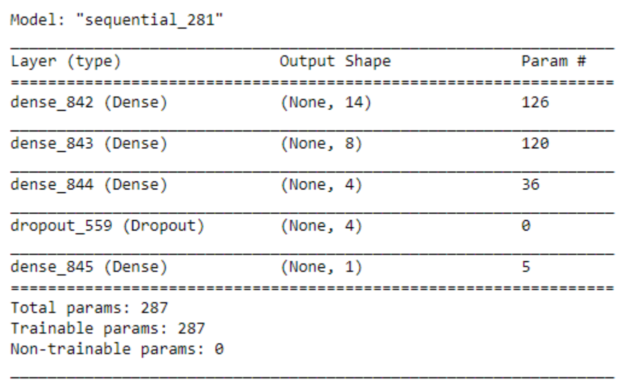


Figure 5: ROC Curve for the linear kernel, gamma = 0.0001, and a C = 0.01

ROC is a probability curve and AUC represents degree or measure of separability, which it shows how well the model is able to distinguish between classes. A higher AUC implies that the model is better at predicting true 0s and 1s. In this case, the higher the AUC, the better the model is at distinguishing between patients with diabetes and no diabetes. It is okay to receive some false positives, since it would lead to the patient to get additional tests where they are hopefully diagnosed as healthy. However, false negatives could be very dangerous. If I predicted a false negative then the patient would miss treatment they might need. Looking at Figure 4, there are many patients that would be flagged as being non-diabetic when they actually are diabetic. This suggests that other factors besides glucose and BMI could have an impact on whether someone is diabetic or not. Future models could use the diabetes pedigree function which is essentially its own model that scores the likelihood of diabetes based on family history. By incorporating genetics/pedigree into the model, it might be able to distinguish the clusters of 1s over top of the 0s.

4. Creating a Neural Network

When making the neural network I plan on using all 8 features to predict the outcome. However, if I remove all rows that have a false zero I narrow down the data greatly, so I will input all the data I have. Furthermore, for the same reasons as the SVM model, I will scale my data. Since the goal is to predict a binary output, I will use the relu and sigmoid activation functions because they exist between 0 and 1 (like the Outcome data). For the MNIST dataset we used a softmax function, however, since I only have two classes (0 and 1s), I should end on a sigmoid function. The validation accuracy of the untuned neural networked was 0.714, worse than our SVM.


```
Model: "sequential_281"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_842 (Dense)            (None, 14)                126
_________________________________________________________________
dense_843 (Dense)            (None, 8)                 120
_________________________________________________________________
dense_844 (Dense)            (None, 4)                 36
_________________________________________________________________
dropout_559 (Dropout)        (None, 4)                 0
_________________________________________________________________
dense_845 (Dense)            (None, 1)                 5
=================================================================
Total params: 287
Trainable params: 287
Non-trainable params: 0
_________________________________________________________________
```

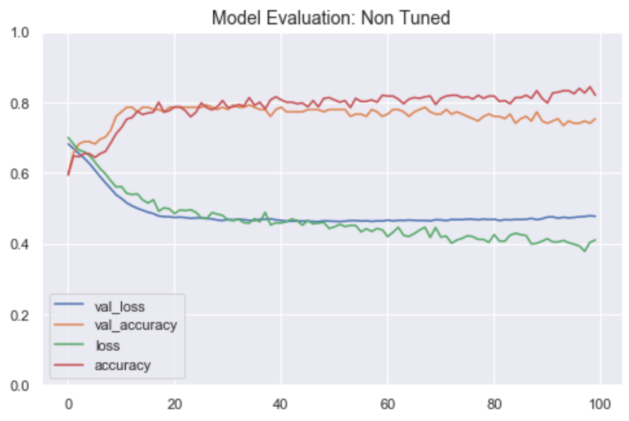
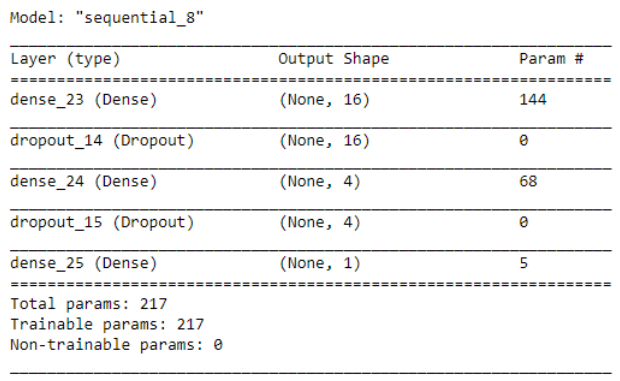


Figure 6: Summary and evaluation of non-turned neural network

I chose to use dense layers because there should be an association between the features, and I used dropout layers for regularization. Dropout layers can help models avoid overfitting by essentially dropping an input neuron when training the model [1]. Since I am allowed to control the percentage of the dropout there should not be any harm in adding it to my model, since parameter tuning could yield a 0 dropout. The dropout rate was partially responsible for the "jitteriness" of the accuracy curves. Using a very similar model to Figure 6 as the framework for my neural network, I hypertuned the number of neurons, learning rate, and dropout rate, resulting in Figure 7.


```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_23 (Dense)             (None, 16)                144
_________________________________________________________________
dropout_14 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_24 (Dense)             (None, 4)                 68
_________________________________________________________________
dropout_15 (Dropout)         (None, 4)                 0
_________________________________________________________________
dense_25 (Dense)             (None, 1)                 5
=================================================================
Total params: 217
Trainable params: 217
Non-trainable params: 0
_________________________________________________________________
```

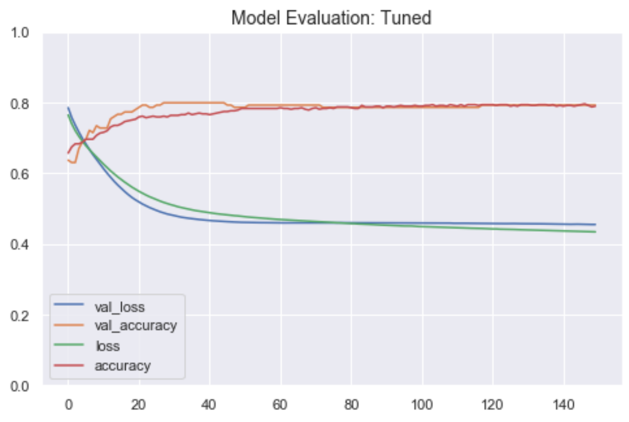


Figure 7: Summary and evaluation of turned neural network

My first layer had a neuron size of 16, second neuron size was 4, with a dropout rate of 0 and learning rate of 0.1. This resulted in an accuracy of about .74, there were minor fluctuations between runs. The summary of the model can be seen in Figure 7. The tuned model produced a smoother looking learning curve and almost has no underfitting or overfitting since both accuracy's are similar. Furthermore, the training and testing accuracy's nearly converge on top of each other rather than diverge.

5. Conclusion

Both the SVM model and the neural network model predicted whether or not someone has diabetes based on a variety of features with similar accuracy. The SVM model received an accuracy of 0.802 and my neural network had an accuracy of .74. Both models have their pros, but with larger data sets the multi-layer neural network it allows for the easy feeding of features. With SVM models, feature extraction becomes more important rather than feeding everything into a model. Furthermore, clean data is important in any model. If there was more time, I would love to play around with the NN and try a variety of neuron sizes and different optimizers, especially if there was bigger data sets. In addition, it would be interesting to run the NN using warm starts. Overall, I would take the NN model.

References

[1] Gero, Aurelien, *Hands-On Machine learning with Scikit-Learn, Keras and TensorFlor*, Seond Edition. O'Reilly Media, 2019.

[2] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., Johannes, R.S. (1988). *Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceed-ings of the Symposium on Computer Applications and Medical Care* (pp. 261 - 265). IEEE Computer Society Press.