Intro To Data Science
Final Project Report

PIMA Diabetes Dataset

Rashmi Pai
February 21, 2021

Abstract

Diabetes is a disease that occurs when your blood glucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energy

Without ongoing, careful management, diabetes can lead to a buildup of sugars in the blood, which can increase the risk of dangerous complications, including stroke and heart disease.

Different kinds of diabetes can occur, and managing the condition depends on the type. Not all forms of diabetes stem from a person being overweight or leading an inactive lifestyle. In fact, some are present from childhood.

Background

The PIMA Diabetes[1] dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Data

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Column	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration a 2 hours in an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skin fold thickness (mm)
Insulin	2-Hour serum insulin (mu U/ml)
BMI	Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction	Diabetes pedigree function
Age	Age (years)
Outcome	Class variable (0 or 1)

Methods

As part of exploration of this dataset, I've chosen four main research topics

1. Can we use Neural Network model to confirm the connection between model complexity and accuracy?

2. Can we use Regression to do some feature importance extraction, perhaps the above MLP model will perform better by filtering out less important features?

3. How do other models compare? If we chose Support Vector Models to do classification, what would be the optimal C value?

4. Does using Ensemble technique help us with a model with better accuracy?

In order to answer my research topics, I decided to look into each feature carefully trying to fix the data range. I also applied scaling techniques to iterate over model accuracy while trying to retain or improve the precision and recall of the models.

In some cases, I'd to compare the outcome of different training ensembles to decide if my technique was better equipped to handle the data at my hand.

Lastly I also did some feature engineering by applying Regression and RandomForests to identify columns that greatly influenced our model accuracies. Furthermore by comparing the feature importance output from different models I came to conclusion that some features always held most influence over others, while some fluctuated depending on the underlying model generation technique.

Results

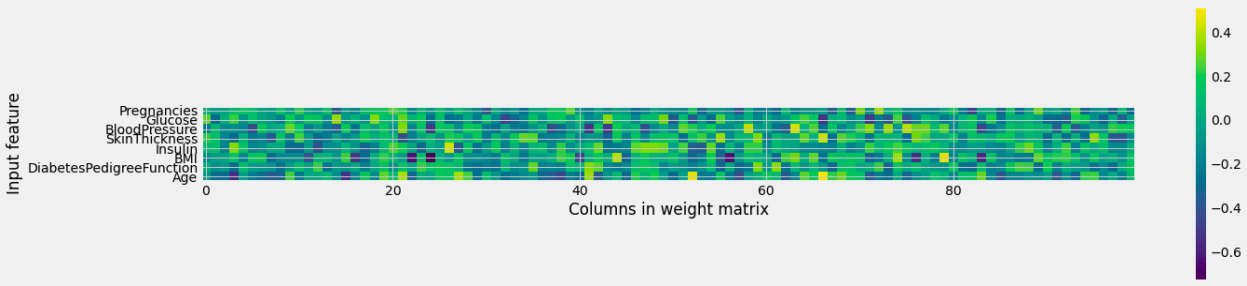
1. Can we use Neural Network model to confirm the connection between model complexity and accuracy[2]?

After analyzing the dataset, I found *Outcome* column to be the best categorical column while the other columns were used as quantitative columns. I used the *sklearn.neural_network.MLPClassifier* to train a model on a split on the dataset.

	precision	recall	f1-score	support
0	0.691176	0.940000	0.796610	100.000000
1	0.666667	0.222222	0.333333	54.000000
accuracy	0.688312	0.688312	0.688312	0.688312
macro avg	0.678922	0.581111	0.564972	154.000000
weighted avg	0.682582	0.688312	0.634162	154.000000

Following was the *classification_report* generated for the trained model.

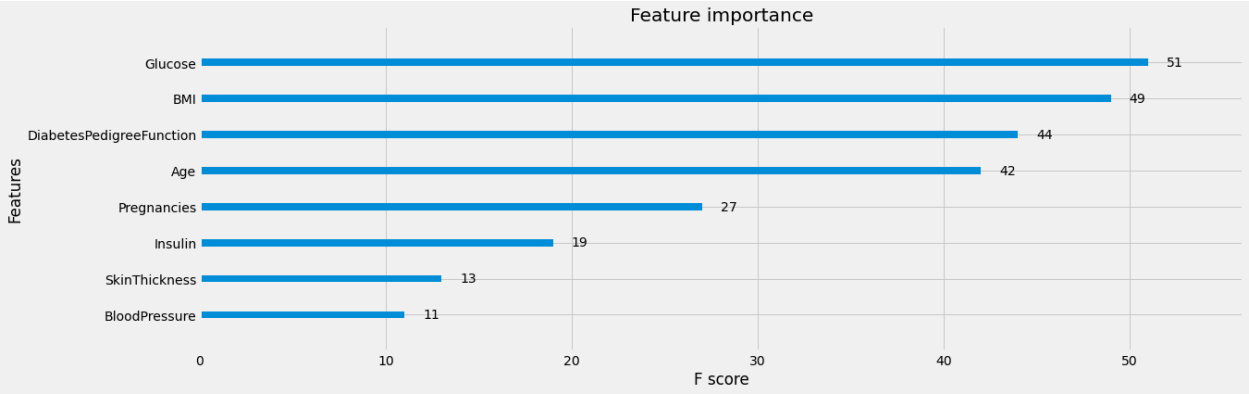
The accuracy of 0.69 with a 69% precision and a high 94% recall is not good enough. So I tried to improve the accuracy of the model by trying to scale out my training dataset. I saw an immediate improvement in the model accuracy of 0.80. I tried to identify which features were important but I couldn't immediately point out which of them held lower weights compared to others as seen below.

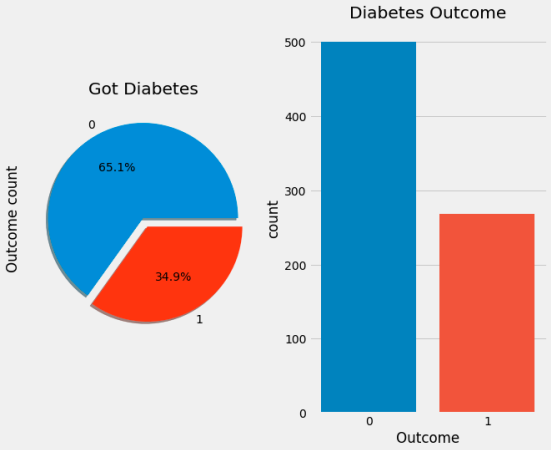


2. Can we use Regression to do some feature importance extraction, perhaps the above MLP model will perform better by filtering out less important features?

As I saw my Neural Network model only could provide a max accuracy rate of 0.80, I tried to then focus on improving its accuracy by only using columns that could greatly influence it by running a series of Regression model fits on my datasets.

By using the embedded and wrapper method, I was able to arrive at following conclusion: Glucose, BMI, DiabetesPedigreeFunction, Age and Pregnancies were clearly more important than Insulin, SkinThickness, BloodPressure.





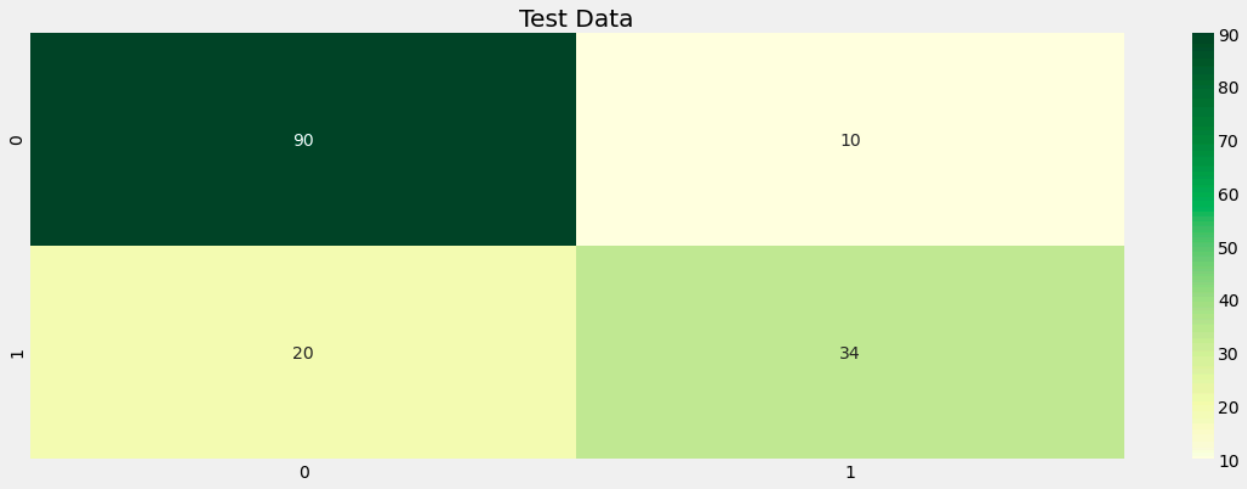
As looked more into the dataset, I could easily see that the data is skewed more towards negative Outcome of diabetes so it only made sense that columns like Insulin, SkinThickness and BloodPressure, didn't influence the models accuracy for positive outcomes.

So I tried to use this new bit of information to re-train my NN model to see if it helped increase the model accuracy and it definitely helped. The models accuracy went up, as did the precision and recall. As compared to earlier accuracy of 0.80, now the model could accurately predict the outcome with 0.86 accuracy. The 86% precision with 94% recall definitely helped the cause.

```
print(classification_report(test_Y, mlp.predict(test_X),
                           digits = 4,
                           target_names=["Not Diabetes",
                                         "Diabetes"]))
```

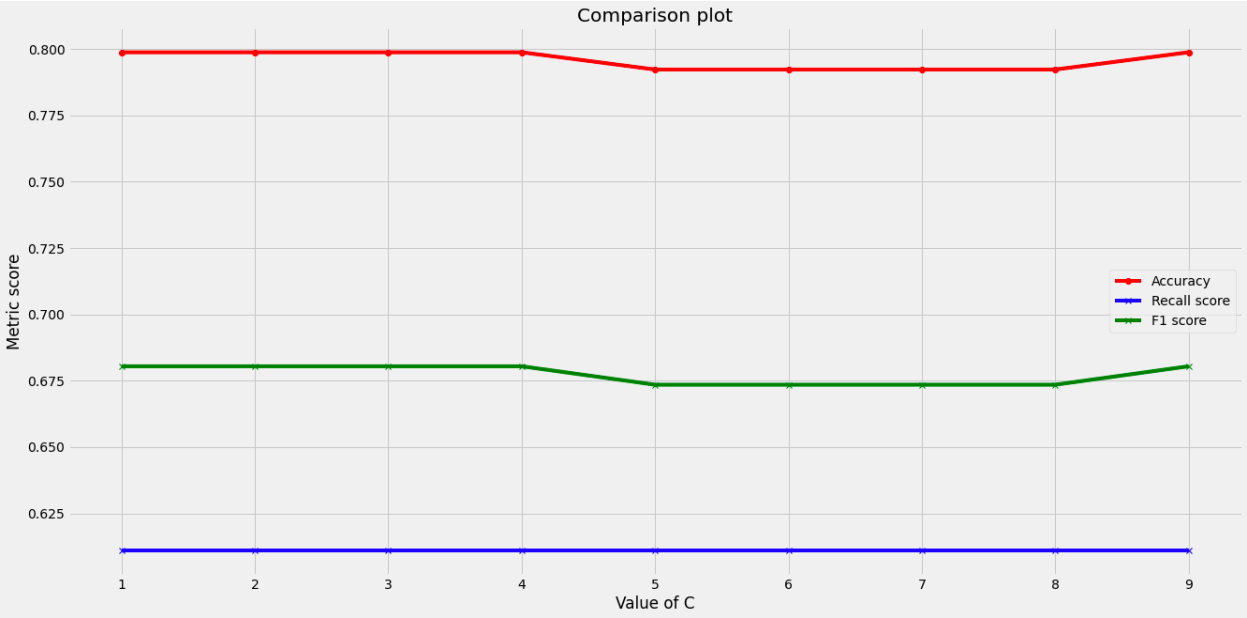
	precision	recall	f1-score	support
Not Diabetes	0.8624	0.9400	0.8995	100
Diabetes	0.8667	0.7222	0.7879	54
accuracy			0.8636	154
macro avg	0.8645	0.8311	0.8437	154
weighted avg	0.8639	0.8636	0.8604	154

Below is the confusion matrix for the trained model.

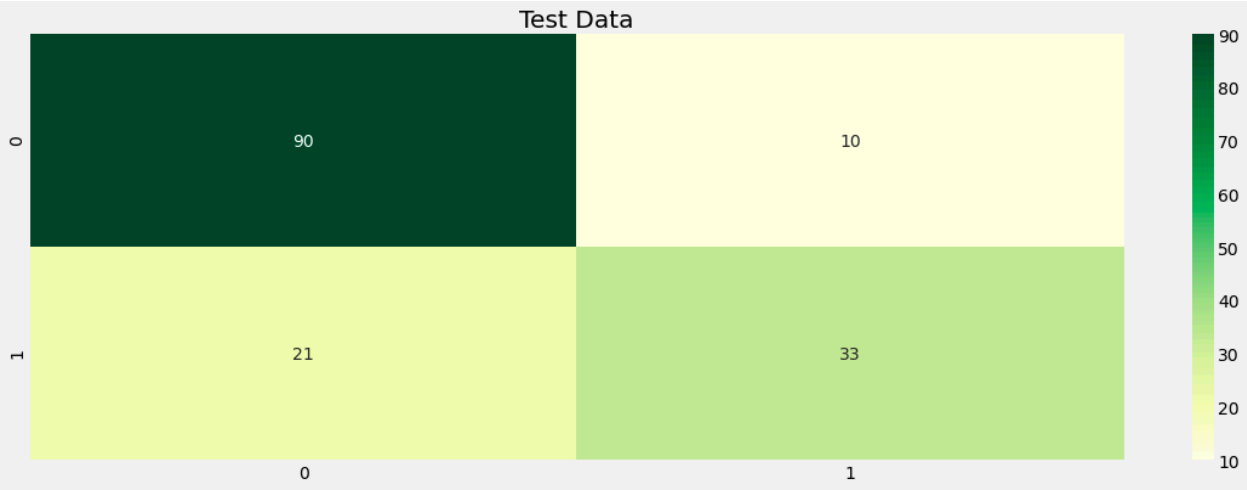


3. How do other models compare? If we chose Support Vector Models to do classification, what would be the optimal C value?

I tried to used Support Vector Classifier on my dataset and it provided a 0.77 accuracy rate without any scaling and 0.80 accuracy with scaling. So in comparison to my NN model the SV-Classifier performed more or less similar. I was curious to find the optimal C value[3] for my SVC model so I ran some iteration and I found that C=1 was the optimal.

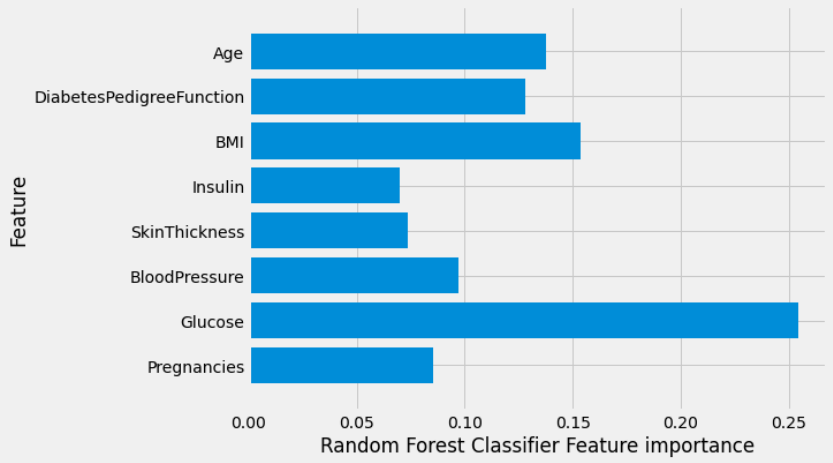


Filtering out features with less weight had no effect on accuracy of the svc model. The confusion matrix for svc model is as shown below:



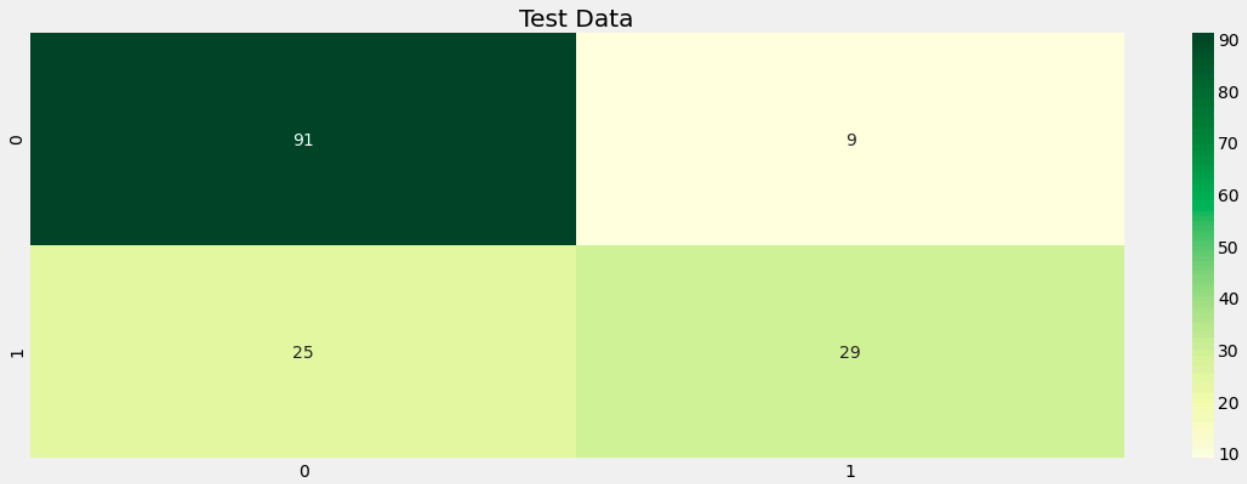
4. Does using Ensemble techniques help us with a model with better accuracy?

I used RandomForestClassifier ensemble as its very easy and quick to generate the feature importance based on the trained model. Following were the observations



As with the LogisticRegression models, Glucose, BMI, Age and DiabetesPedgreeFunction seem to have higher weights as compared to Insulin, Blood Pressure and SkinThickness.

This is consistent with data we have as negative Outcomes outweigh the positive Outcomes. The confusion matrix for the trained RandomForestClassifier is as shown below.



Finally I tried to compare RFC to other ensemble techniques using the cross_val_score and plotting the mean accuracy each model had to offer and this was the outcome.

RandomForest was not the best but also not the worst when it came to accuracy among other ensemble techniques.

```
from sklearn.model_selection import cross_val_score,KFold

for clf, clf_name in zip(classifiers, classifier_names):
    cv_scores = cross_val_score(clf, train_X,train_Y, cv=5)

    print(clf_name, ' mean accuracy: ',
          round(cv_scores.mean()*100, 3), '% std: ',
          round(cv_scores.var()*100, 3),'%')
```

```
KNN  mean accuracy:  70.524 % std:  0.037 %
Decision Tree  mean accuracy:  68.235 % std:  0.062 %
SVC  mean accuracy:  72.482 % std:  0.04 %
SVC with sigmoid kernel  mean accuracy:  72.482 % std:  0.04 %
Gaussian Naive Bayes  mean accuracy:  77.201 % std:  0.039 %
RidgeClassifier  mean accuracy:  73.78 % std:  0.016 %
RandomForest  mean accuracy:  72.961 % std:  0.077 %
```

Discussions and Conclusions

As seen from the above Methods and Results, I saw that because the dataset is skewed towards negative Outcome, the features, which in real world play an important level in determining whether a person has diabetes, were heavily down weighted. Furthermore, for some models, doing feature selection and engineering does help improve its accuracy while for some it doesn't. The ensemble techniques I applied were fairly coherent to my Regression feature engineering method.

As a future scope, it would be nice to see the effect of a more proportionate data equally biased towards both positive and negative outcomes.

References

[1] Learning, U. (2016, October 06). Pima Indians Diabetes Database. Retrieved February 22, 2021, from https://www.kaggle.com/uciml/pima-indians-diabetes-database

[2] Complete guide to artificial neural network concepts & models. (n.d.). Retrieved February 22, 2021, from https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/

[3] Support-vector machine. (2021, February 18). Retrieved February 22, 2021, from https://en.wikipedia.org/wiki/Support-vector_machine#Soft-margin