EDA and Regression Coursework

Student ID: 10724837

1 The Data

The data has 9 columns and 750 entries. 8 columns are the given predictors and the final column is Outcome, which equals 1 if the woman eventually ends up with diabetes, 0 otherwise.

There are no null or negative values in the dataset. There are, however, some unreasonably large values which will treat as outliers in our analysis.

There are zero values in columns where there should not be; namely, the Glucose, BloodPressure, SkinThickness, Insulin and BMI columns. This poses a problem because it is not clear why zero values were used, instead of leaving the entries blank. For example, whilst a zero in the BMI column is clearly incorrect, there is no way to tell whether a zero in the Pregnancies column is a true zero, or if it was a missing record.

2 Exploratory Data Analysis

I complete the EDA within Python.

First, we must deal with the zero values mentioned earlier. In some columns, such as Glucose, BloodPressure or BMI, there are only a few zero values, so imputing these values will have minimal effect on our overall analyses. SkinThickness and Insulin, on the other hand, have 221 and 362 zeroes respectively, out of the 750 total observations. Imputing these values will have a non-negligible impact on our results.

Dropping SkinThickness and Insulin would have been a valid option in this situation, and one which was strongly considered. As a general rule of thumb, columns should be dropped entirely if they have over 50% incorrect or missing values. Since these columns do not satisfy this condition, we keep them in and impute them instead.

Before deciding how to impute, we look at how each column is distributed. We see that if we ignore the zero values, each of these columns somewhat resemble normal distributions. Hence, mean value imputation will be a decent choice here.

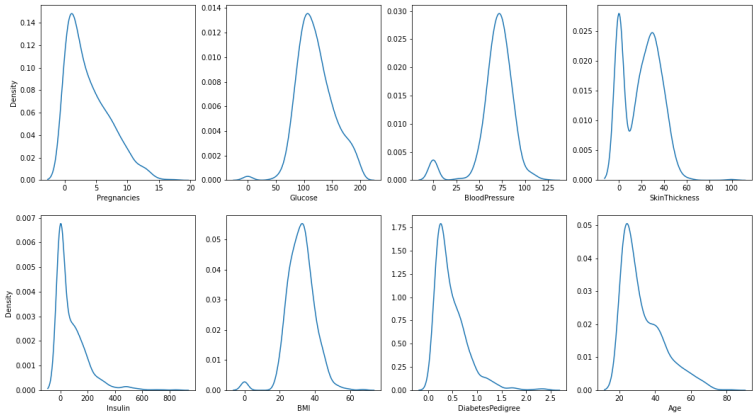


Figure 1: The distributions for each predictor.

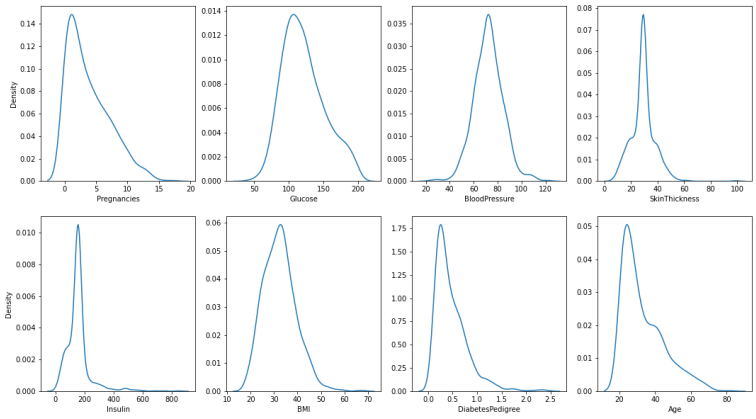


Figure 2: The distributions for each predictor, after applying mean value imputation.

Some of the distributions are positively skewed, particularly Pregnancies, DiabetesPedigree and Age. A log or square root transform on these variables will reduce their skewness and improve later models. Since Pregnancies has zero values, only the square root transform will work. For DiabetesPedigree and Age, we can apply both and see the results.

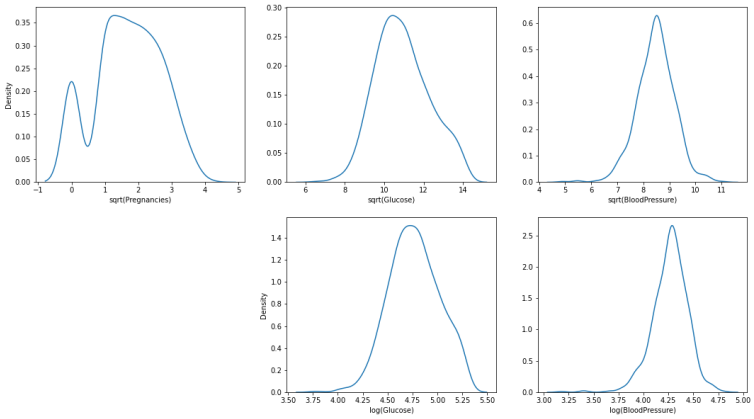


Figure 3: The distributions on applying square root or log transformations.

The square root transformation made Pregnancies bimodal, so we will leave Pregnancies as is. Both transforms worked well on DiabetesPedigree and Age, so later on we will consider adding the transformed variables to our model.

In order to deal with outliers, we look at the boxplots of each column first.

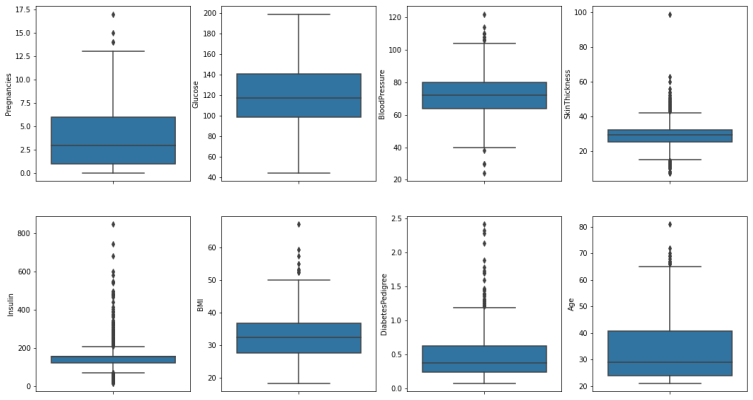


Figure 4: The boxplots for each predictor.

Note that the applied imputations has rendered some of the boxplots somewhat useless. Each column has different numbers of outliers, and we do not want to

remove too much data. Although outlier removal methods such as Z-score or IQR removal are more methodical, in our case, it is more sensible to observe the boxplots and remove outliers using our own intuition. We go ahead and remove the obvious outliers, 10 in total from 5 of the columns.

Checking the correlations between the variables, there are no outstandingly high values, so no certain conclusions can be made. We can, however, make some very useful deductions. BMI and SkinThickness, and Age and Pregnancies, are two pairs of predictors with high correlations (0.564042 and 0.555447 respectively). A higher correlation between two predictors implies that only one or the other will be included in a final regression model. Glucose has a higher correlation with Outcome than the other predictors (0.489844), which implies that it is more likely to be included in our final model. BloodPressure has the lowest correlation with Outcome (0.164569), so it is less likely to be included.

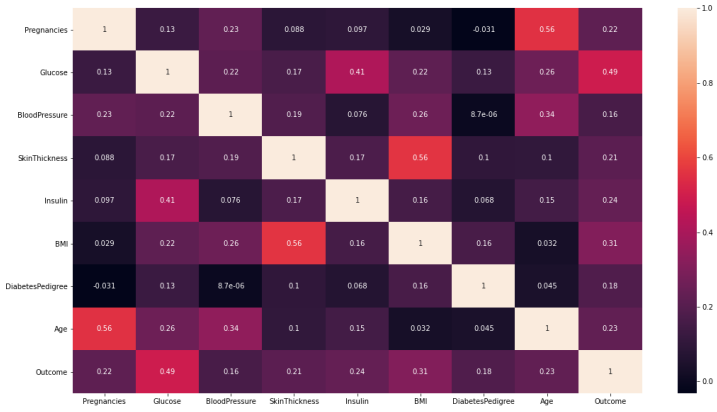


Figure 5: The correlation heatmap.

The following boxplots show the impact each predictor has on Outcome. The only one of note is BloodPressure, which we can see does not change much between the two outcomes (this is reflective of the low correlation between them). Again, this reinforces BloodPressure as a candidate for removal.

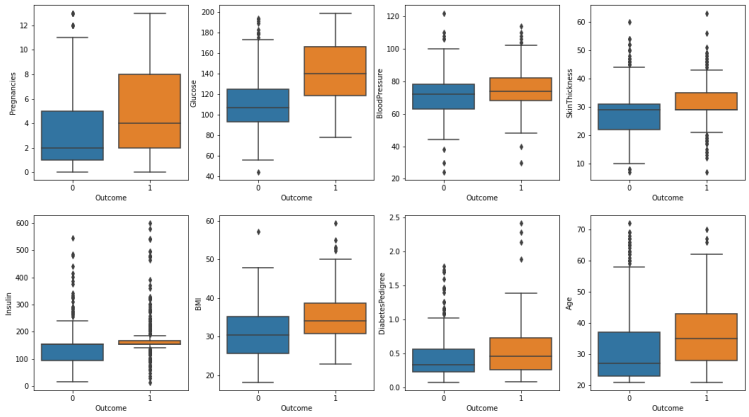


Figure 6: The boxplots comparing predictors between the women who do and do not end up with diabetes.

The pairplots are somewhat distorted due to the amount of imputation carried out above. They do not tell us anything of particular importance here.

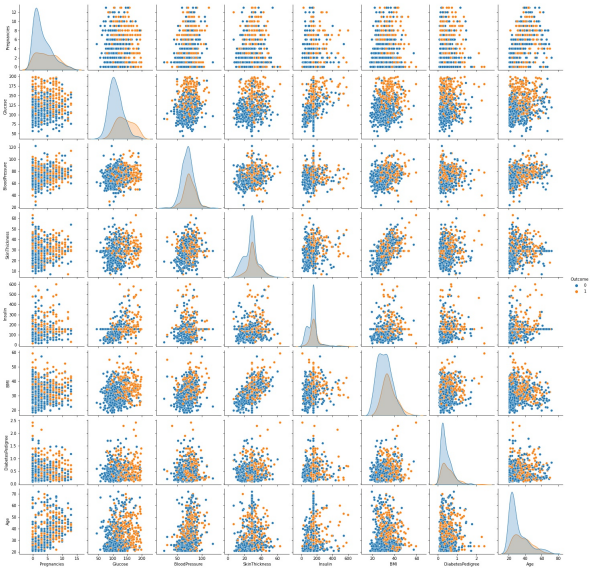


Figure 7: The pairplots for all predictors.

3 SevenOrMorePregnancies

I add the SevenOrMorePregnancies column in Python, save the edited .csv file, then switch to R for the rest of the analysis.

Since Outcome is a binary categorical variable, the appropriate choice is a logistic regression model, with formula $Outcome \sim SevenOrMorePregnancies$.

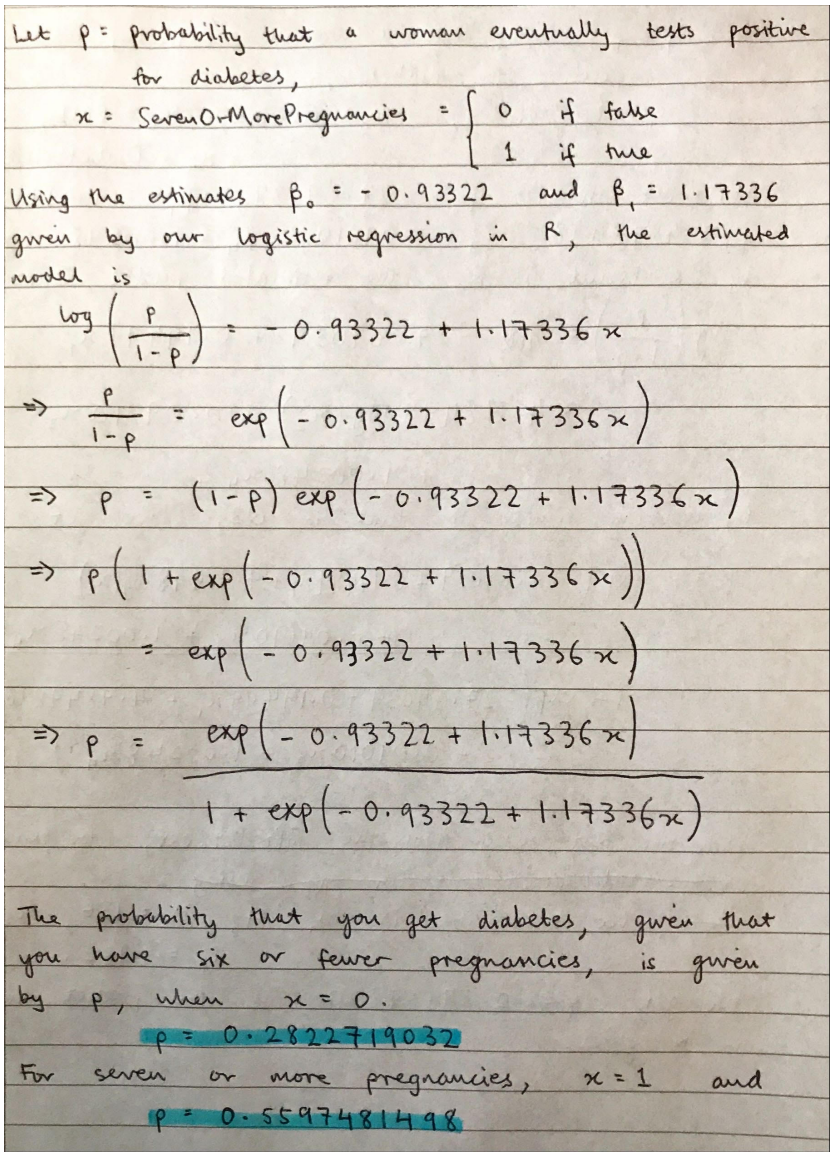


Figure 8: The working out for the requested probabilities.

4 Final Regression Model

As before, we use a logistic regression model. When developing a regression model with multiple predictors, we attempt to pick the predictors which best describe the dependent variable, whilst keeping the model as simple as possible. For the comparison of logistic regression models, a useful statistic is each model's AIC. It is calculated using the MLE of the model, but also penalises the inclusion of more independent variables. We want to lower the AIC with each iteration of the model, without adding too many predictors, so that we avoid overfitting and keep model complexity down.

To do this, we use a method known as "backward stepwise regression", where we start with a model including all predictors, then eliminate the least statistically significant predictor, one-by-one.

With all predictors, we get an AIC of 700.69. The least significant predictor was Insulin, with a p-value of 0.85859.

Removing Insulin and trying again, we get an AIC of 698.72, an improvement. Thus, we can quite confidently remove Insulin from the model.

Repeating this process a few times, our model is left with Pregnancies, Glucose, BMI and DiabetesPedigree. At this point, all predictors are statistically significant. Removing any of these leads to noticeable increases in the AIC.

To reinforce confidence in our result, we can use the boot.stepAIC function, which takes samples with replacement from our data, and runs the backwards stepwise regression process automatically on each one. With 50 samples, the same 4 predictors we have were returned every time. Thus, our predictors are quite consistent with our data.

We still have room for improvement, because we have not yet considered a log or square root transform on Glucose, as discussed earlier. If we replace Glucose with sqrt(Glucose) or log(Glucose), we lower the AIC even further. Since log(Glucose) lowers it more, we choose this.

Now that we have finalised our model, here is the working out for our final regression equation, and an example of how we can use it to make our predictions:

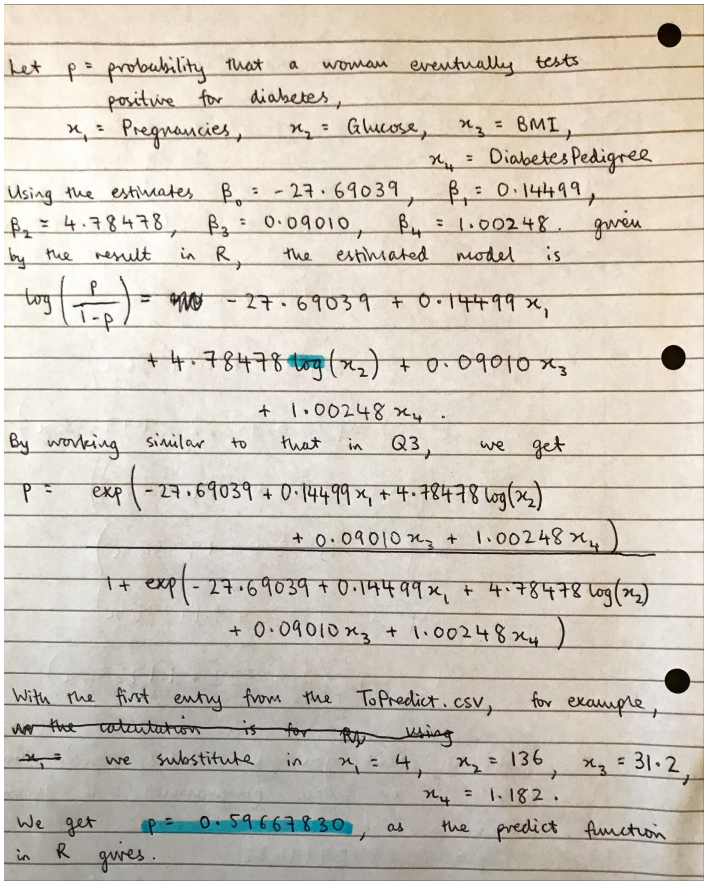


Figure 9: The working out for our final model.

We can complete all the predictions using the predict function in R. We get:

0.57354249, 0.28547417, 0.08934668, 0.82244357, 0.71481715

5 Appendix

On the following pages, I include my Jupyter notebook and R code.

5.1 Python Code

In [1]:
```
########
# Student ID: 10724837
# In this notebook, I carry out some basic observation of the data, as well
# before moving over to R to develop the regression models.
########
```

In [2]:
```
# import packages we will be using
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
```

In [3]:
```
# load the data and observe basic info
diabetes = pd.read_csv("PimaDiabetes.csv")
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 750 entries, 0 to 749
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Pregnancies       750 non-null    int64  
 1   Glucose           750 non-null    int64  
 2   BloodPressure     750 non-null    int64  
 3   SkinThickness     750 non-null    int64  
 4   Insulin           750 non-null    int64  
 5   BMI               750 non-null    float64
 6   DiabetesPedigree  750 non-null    float64
 7   Age               750 non-null    int64  
 8   Outcome           750 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 52.9 KB
```

In [4]:
```
# check for null values
diabetes.isna().sum(axis = 0)
```

Out[4]:
```
Pregnancies         0
Glucose             0
BloodPressure       0
SkinThickness       0
Insulin             0
BMI                 0
DiabetesPedigree    0
Age                 0
Outcome             0
dtype: int64
```

In [5]:
```
# there are no null values!
```

In [6]:
```
# calculate some key summary statistics, mainly interested in the min and mc
diabetes.describe()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabe
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	
mean	3.844000	120.737333	68.982667	20.489333	80.378667	31.959067	
std	3.370085	32.019671	19.508814	15.918828	115.019198	7.927399	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	36.500000	32.000000	
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

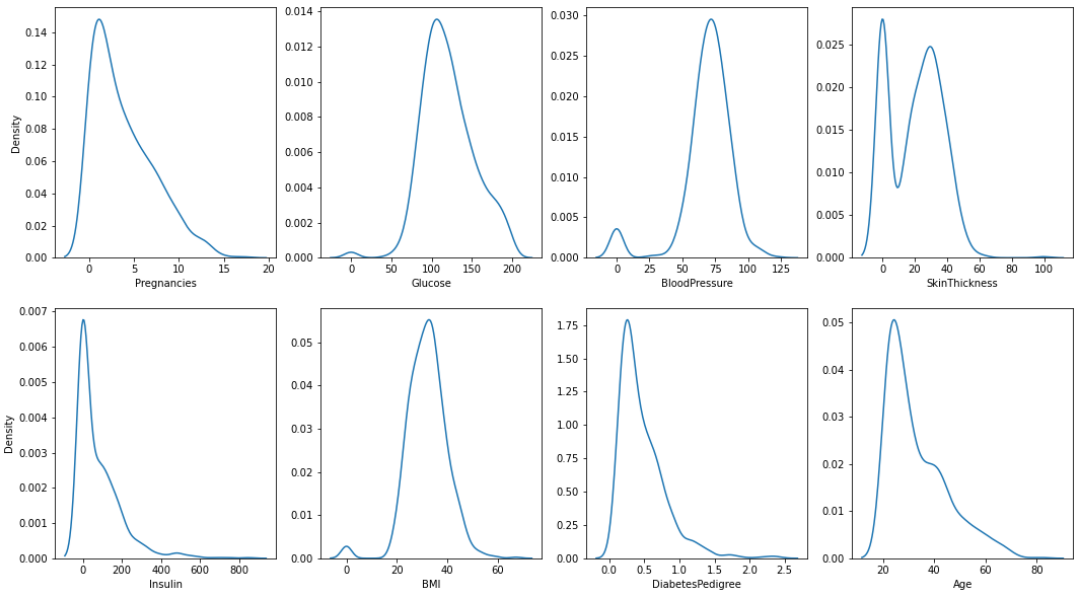
In [7]:
```
# there are no negative min values, a good sign
# there are some very large max values, such as 846 in the Insulin columns
# we will deal with such outliers later
```

In [8]:
```
# check the number of zero values in each column
diabetes.isin([0]).sum(axis = 0)
```

Out[8]:
```
Pregnancies         109
Glucose               5
BloodPressure        35
SkinThickness       221
Insulin             362
BMI                  11
DiabetesPedigree      0
Age                   0
Outcome             490
dtype: int64
```

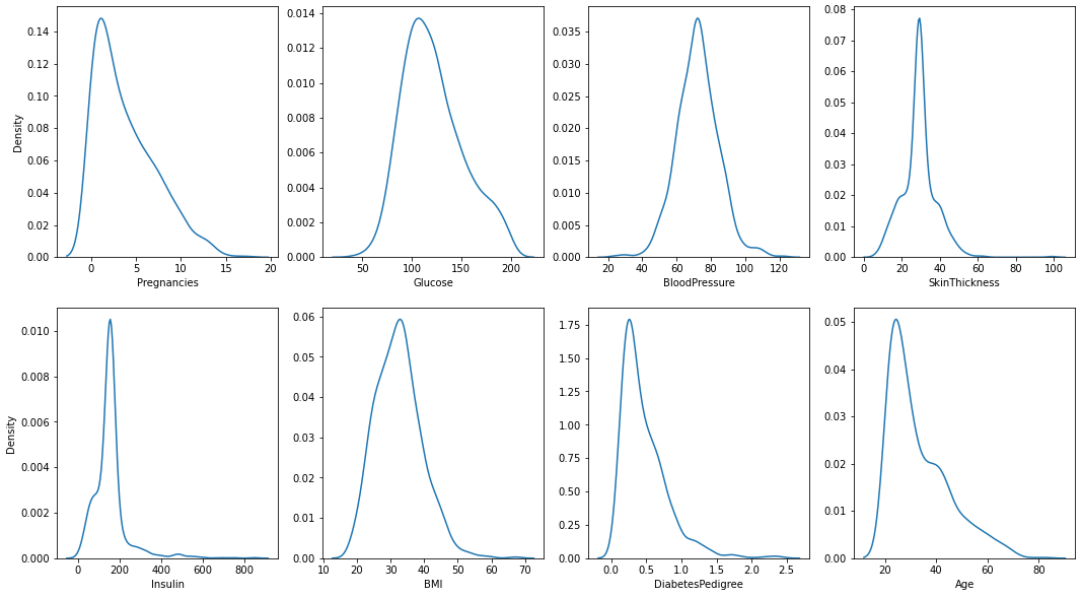
In [9]:
```
# there are lots of zeroes in columns where it doesn't make sense
# for example, you definitely cannot have a SkinThickness or BMI of 0
# the columns which have zeroes when they should not are: Glucose, BloodPres
# this is a problem with the data and we need to look at how these 0s are di
```

In [10]:
```
# plot the distributions of each column
plt.figure(figsize = (18,10),facecolor=(1, 1, 1))

plt.subplot(2,4,1)
sns.kdeplot(data = diabetes['Pregnancies'])

plt.subplot(2,4,2)
sns.kdeplot(data = diabetes['Glucose'])
plt.ylabel("")

plt.subplot(2,4,3)
sns.kdeplot(data = diabetes['BloodPressure'])
plt.ylabel("")

plt.subplot(2,4,4)
sns.kdeplot(data = diabetes['SkinThickness'])
plt.ylabel("")

plt.subplot(2,4,5)
sns.kdeplot(data = diabetes['Insulin'])

plt.subplot(2,4,6)
sns.kdeplot(data = diabetes['BMI'])
plt.ylabel("")

plt.subplot(2,4,7)
sns.kdeplot(data = diabetes['DiabetesPedigree'])
plt.ylabel("")

plt.subplot(2,4,8)
sns.kdeplot(data = diabetes['Age'])
plt.ylabel("")

plt.savefig(fname = "diabetesfig1")
```



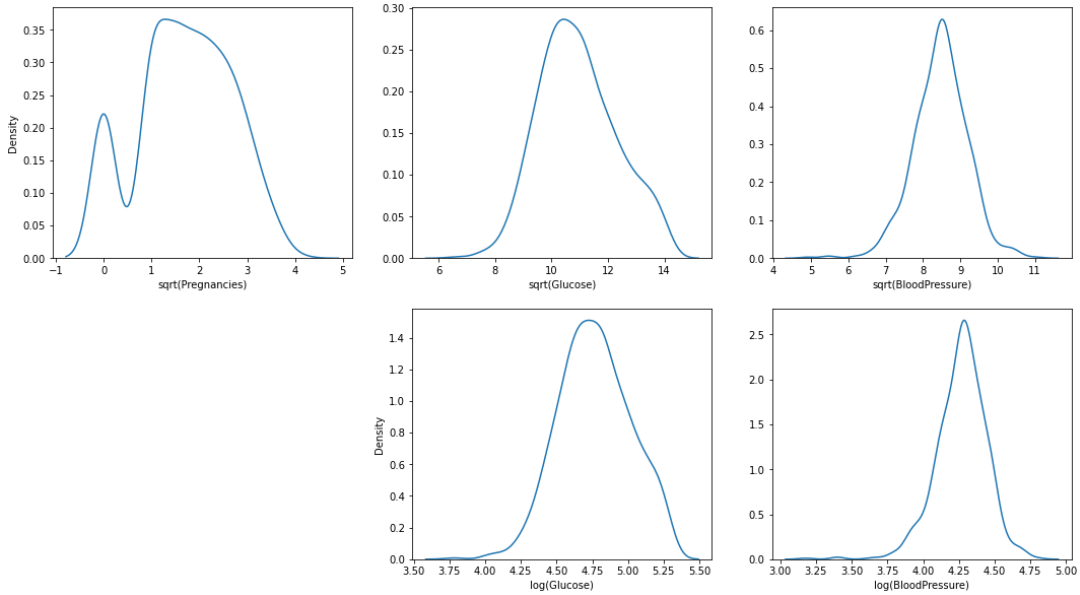
In [11]:
```
# ignoring the zero values, these columns resemble normal distributions
# use mean value imputation to replace all zero values
# apply the mean value imputations
meanImputationColumns = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insul
for column in meanImputationColumns:
    mean = diabetes[diabetes[column] != 0][column].mean()
    diabetes[column] = diabetes[column].replace(0, mean)
```

In [12]:
```
# observe the distributions now that we have imputed zero values
plt.figure(figsize = (18,10),facecolor=(1, 1, 1))

plt.subplot(2,4,1)
sns.kdeplot(data = diabetes['Pregnancies'])

plt.subplot(2,4,2)
sns.kdeplot(data = diabetes['Glucose'])
plt.ylabel("")

plt.subplot(2,4,3)
sns.kdeplot(data = diabetes['BloodPressure'])
plt.ylabel("")

plt.subplot(2,4,4)
sns.kdeplot(data = diabetes['SkinThickness'])
plt.ylabel("")

plt.subplot(2,4,5)
sns.kdeplot(data = diabetes['Insulin'])

plt.subplot(2,4,6)
sns.kdeplot(data = diabetes['BMI'])
plt.ylabel("")

plt.subplot(2,4,7)
sns.kdeplot(data = diabetes['DiabetesPedigree'])
plt.ylabel("")

plt.subplot(2,4,8)
sns.kdeplot(data = diabetes['Age'])
plt.ylabel("")

plt.savefig(fname = "diabetesfig2")
```



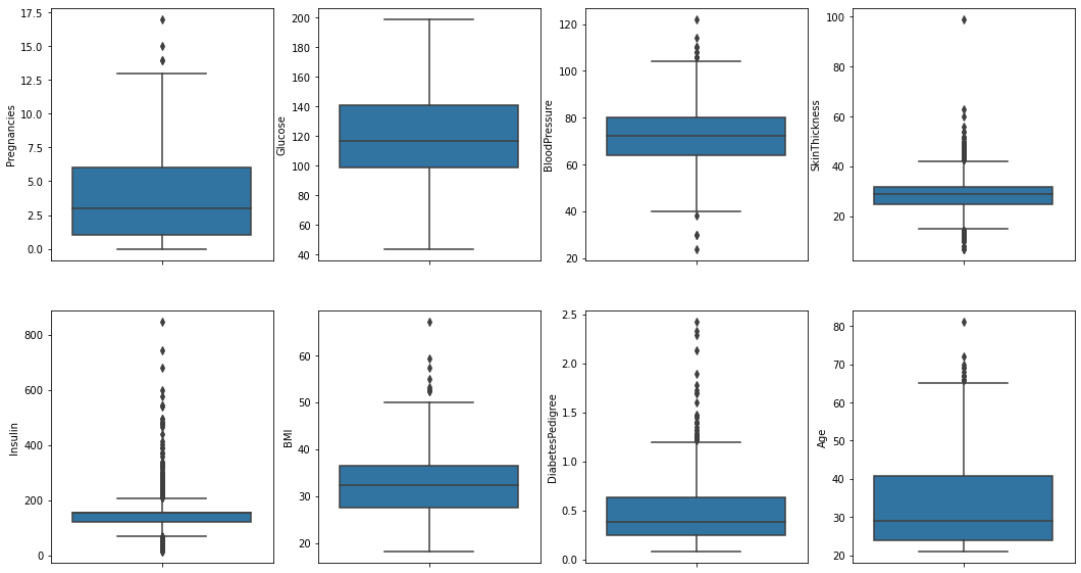
In [13]:
```
# some of the distributions are right-skewed, particularly Pregnancies, Diab
# to fix this, we can use either log or square root transformations
# since Pregnancies has zero values, log transform will not work
# as for DiabetesPedigree and Age, we can try both log and square root trans
```

In [14]:
```
# plot the square root transforms of all three columns
plt.figure(figsize = (18,10),facecolor=(1, 1, 1))

plt.subplot(2,3,1)
sns.kdeplot(data = np.sqrt(diabetes['Pregnancies']))
plt.xlabel("sqrt(Pregnancies)")

plt.subplot(2,3,2)
sns.kdeplot(data = np.sqrt(diabetes['Glucose']))
plt.xlabel("sqrt(Glucose)")
plt.ylabel("")

plt.subplot(2,3,3)
sns.kdeplot(data = np.sqrt(diabetes['BloodPressure']))
plt.xlabel("sqrt(BloodPressure)")
plt.ylabel("")


# plot the log transforms of Glucose and BloodPressure
plt.subplot(2,3,5)
sns.kdeplot(data = np.log(diabetes['Glucose']))
plt.xlabel("log(Glucose)")

plt.subplot(2,3,6)
sns.kdeplot(data = np.log(diabetes['BloodPressure']))
plt.xlabel("log(BloodPressure)")
plt.ylabel("")


plt.savefig(fname = "diabetesfig3")
```



In [15]:
```
# the square root transform made Pregnancies bimodal, so leave Pregnancies w
# both sqrt and log transforms worked well on Glucose and BloodPressure
# so consider using log(Glucose) and log(BloodPressure) in our later models
```

In [16]:
```
# we move onto dealing with outliers
# plot boxplots to see how outliers are distributed visually
plt.figure(figsize = (18,10),facecolor=(1, 1, 1))

plt.subplot(2,4,1)
sns.boxplot(y = diabetes['Pregnancies'])

plt.subplot(2,4,2)
sns.boxplot(y = diabetes['Glucose'])

plt.subplot(2,4,3)
sns.boxplot(y = diabetes['BloodPressure'])

plt.subplot(2,4,4)
sns.boxplot(y = diabetes['SkinThickness'])

plt.subplot(2,4,5)
sns.boxplot(y = diabetes['Insulin'])

plt.subplot(2,4,6)
sns.boxplot(y = diabetes['BMI'])

plt.subplot(2,4,7)
sns.boxplot(y = diabetes['DiabetesPedigree'])

plt.subplot(2,4,8)
sns.boxplot(y = diabetes['Age'])

# save the figure
plt.savefig(fname = "diabetesfig4")
```



In [17]:
```
# z-score or IQR methods would be more methodical to remove outliers, but sc
# and some have barely any
# we do not want to remove too much data, so remove outliers by observation

# remove Pregnancies above 13
# define the condition for removal
condition = diabetes['Pregnancies'] > 13
# remove the entries where this is true
diabetes = diabetes[~condition]

# remove the obvious outlier for SkinThickness
condition = diabetes['SkinThickness'] > 80
diabetes = diabetes[~condition]

# remove Insulin above 620
condition = diabetes['Insulin'] > 620
diabetes = diabetes[~condition]

# remove the top outlier for BMI
condition = diabetes['BMI'] > 65
diabetes = diabetes[~condition]

# remove the top outlier for Age
condition = diabetes['Age'] > 80
diabetes = diabetes[~condition]

diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 740 entries, 0 to 749
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Pregnancies       740 non-null    int64
 1   Glucose           740 non-null    float64
 2   BloodPressure     740 non-null    float64
 3   SkinThickness     740 non-null    float64
 4   Insulin           740 non-null    float64
 5   BMI               740 non-null    float64
 6   DiabetesPedigree  740 non-null    float64
 7   Age               740 non-null    int64
 8   Outcome           740 non-null    int64
dtypes: float64(6), int64(3)
memory usage: 57.8 KB
```

In [18]:
```
# we have removed only 10 entries in total, a small amount, but it will impr
```

In [19]:
```
# calculate the covariance matrix
diabetes.cov()
```

Out[19]:

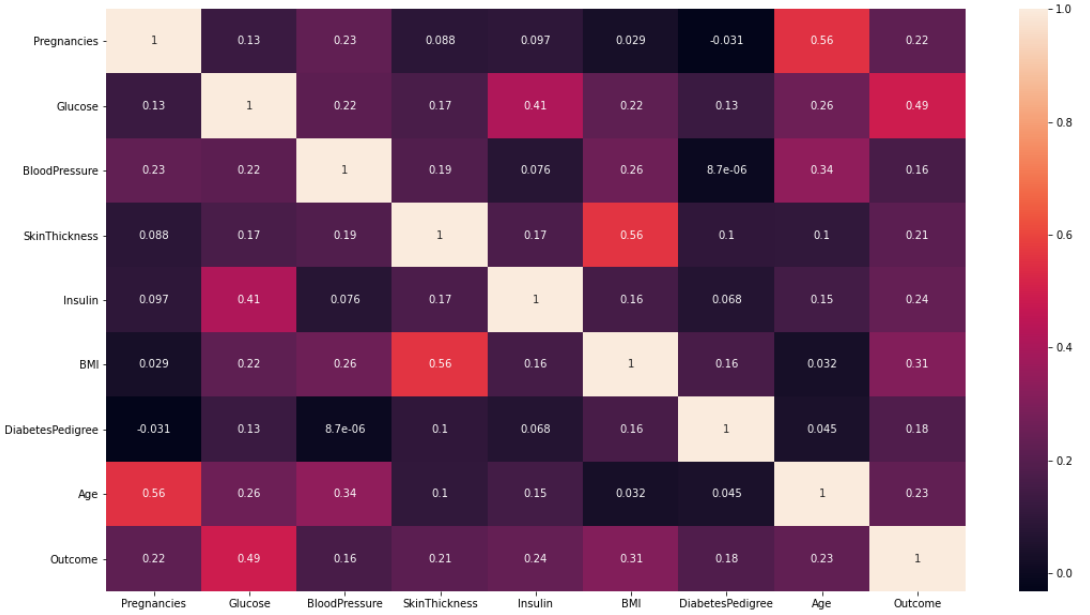
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	10.735272	12.504876	9.082032	2.447829	24.190580	0.64
Glucose	12.504876	908.084944	79.560776	43.060065	945.514805	45.11
BloodPressure	9.082032	79.560776	146.576174	19.311283	70.400334	21.19
SkinThickness	2.447829	43.060065	19.311283	71.356030	112.342561	32.26
Insulin	24.190580	945.514805	70.400334	112.342561	5779.974721	81.16
BMI	0.645843	45.115947	21.199835	32.261255	81.160529	45.84
DiabetesPedigree	-0.033573	1.250934	0.000034	0.278894	1.690199	0.36
Age	20.983482	88.719892	47.644872	9.960717	132.732698	2.51
Outcome	0.341420	7.006599	0.945726	0.847734	8.566895	0.99

In [20]:
```
# calculate the correlation matrix
diabetes.corr()
```

Out[20]:

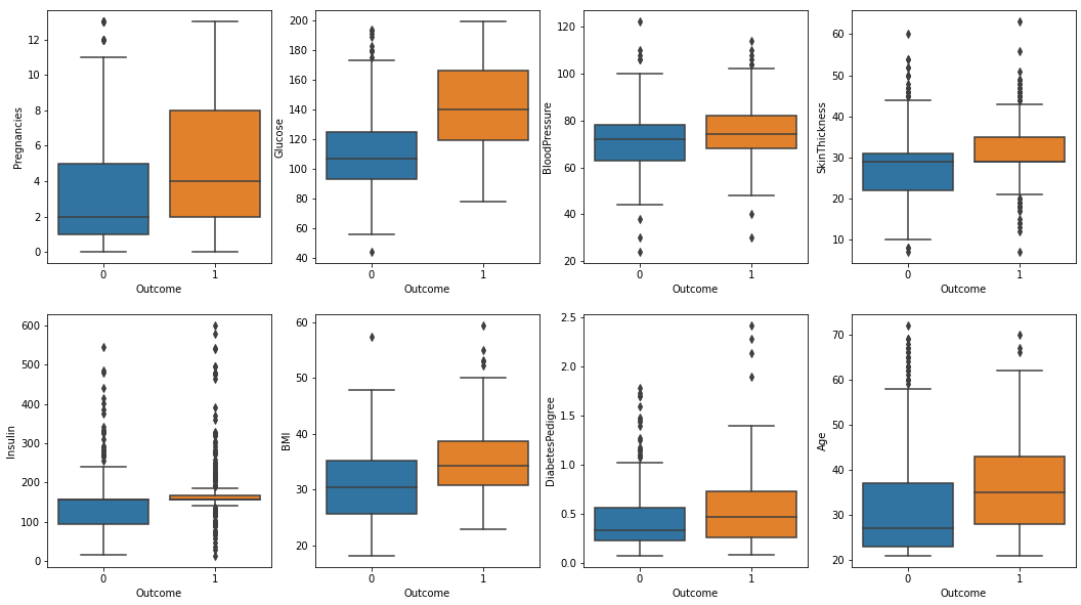
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
Pregnancies	1.000000	0.126651	0.228952	0.088442	0.097113	0.029112
Glucose	0.126651	1.000000	0.218074	0.169159	0.412707	0.221112
BloodPressure	0.228952	0.218074	1.000000	0.188827	0.076486	0.258611
SkinThickness	0.088442	0.169159	0.188827	1.000000	0.174931	0.564042
Insulin	0.097113	0.412707	0.076486	0.174931	1.000000	0.157662
BMI	0.029112	0.221112	0.258611	0.564042	0.157662	1.000000
DiabetesPedigree	-0.031372	0.127095	0.000009	0.101084	0.068066	0.162814
Age	0.555447	0.255346	0.341315	0.102270	0.151421	0.032204
Outcome	0.219531	0.489844	0.164569	0.211426	0.237396	0.308122

In [21]:
```
# alternatively, show as a heatmap
plt.figure(figsize = (18,10),facecolor=(1, 1, 1))
sns.heatmap(data = diabetes.corr(), annot = True)

plt.savefig(fname = "diabetesfig5")
```



In [22]:
```
# the strongest correlations are between BMI and SkinThickness and Age and P
# none of these correlations are strong enough to instantly remove anything,
# are less likely to keep these pairs in the model together
# Glucose has a higher correlation with Outcome than the other predictors, s
# BloodPressure has the lowest, so we are less likely to keep this
```

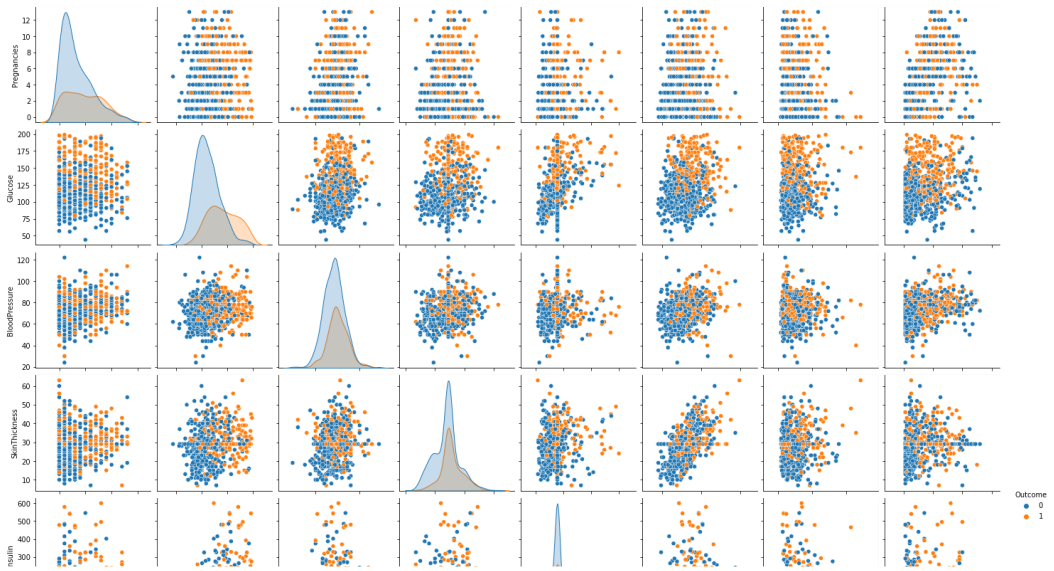
In [23]:
```
# boxplots for the sake of seeing how each predictor impacts the Outcome
plt.figure(figsize = (18,10), facecolor=(1, 1, 1))

plt.subplot(2,4,1)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['Pregnancies'])

plt.subplot(2,4,2)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['Glucose'])

plt.subplot(2,4,3)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['BloodPressure'])

plt.subplot(2,4,4)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['SkinThickness'])

plt.subplot(2,4,5)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['Insulin'])

plt.subplot(2,4,6)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['BMI'])

plt.subplot(2,4,7)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['DiabetesPedigree'])

plt.subplot(2,4,8)
sns.boxplot(x = diabetes['Outcome'], y = diabetes['Age'])

# save the figure
plt.savefig(fname = "diabetesfig6")
```



In [24]:
```
# due to the high number of imputations carried out on Insulin and SkinThick
# BloodPressure does not change very much between the two, which was also re
# again, it is likely a candidate for removal
```

In [25]:
```
# pairplots for all predictors
plt.figure(figsize = (18,10),facecolor=(1, 1, 1))

sns.pairplot(data = diabetes, hue = "Outcome")

plt.savefig(fname = "diabetesfig7.jpg")
```

```
<Figure size 1296x720 with 0 Axes>
```



In [26]:
```
# these pairplots do not show any major patterns, it is not very informative
```

In [27]:
```
# add the column 7 or more pregnancies
diabetes['SevenOrMorePregnancies'] = np.where(diabetes['Pregnancies'] >= 7,
```

In [28]:
```
# save the diabetes dataset as a .csv and move over to R to carry on analysi
diabetes.to_csv("PimaDiabetes2.csv", index = False)
```

5.2 R Code

```
> #########
> # Student ID: 10724837
> # In this R file, I experiment with different regression models,
> # before choosing a final one and using it to predict the outcome
> # of some test data.
> #########
>
>
> # load the data
> diabetes = read.csv("PimaDiabetes2.csv")
> attach(diabetes)
>
>
> # use a logistic regression model because Outcome is categorical with
> # two classes, '0' and '1'
> # fit a regression model with SevenOrMorePregnancies predicting Outcome
> model = glm(data = diabetes, family = binomial,
+             formula = Outcome ~ SevenOrMorePregnancies)
> summary(model)

Call:
glm(formula = Outcome ~ SevenOrMorePregnancies, family = binomial,
    data = diabetes)

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                -0.93322    0.09217 -10.125  < 2e-16 ***
SevenOrMorePregnanciesTrue  1.17336    0.18444   6.362 1.99e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 909.64  on 738  degrees of freedom
AIC: 913.64

Number of Fisher Scoring iterations: 4

>
>
> # again, use a logistic regression model
> # we look to apply backward stepwise regression
> # start with a saturated model, then remove the least significant
```

```
> # predictor one-by-one, trying to lower AIC
> # ignore SevenOrMorePregnancies as this is similar to Pregnancies
> model2 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + Glucose + BloodPressure
+              + SkinThickness + Insulin + BMI + DiabetesPedigree
+              + Age)
> summary(model2)

Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
    SkinThickness + Insulin + BMI + DiabetesPedigree + Age, family = binomial,
    data = diabetes)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)      -9.2526189  0.8434654 -10.970  < 2e-16 ***
Pregnancies       0.1293377  0.0339573   3.809  0.00014 ***
Glucose           0.0368701  0.0039400   9.358  < 2e-16 ***
BloodPressure    -0.0071947  0.0087553  -0.822  0.41122
SkinThickness     0.0052472  0.0137753   0.381  0.70327
Insulin           0.0002374  0.0013323   0.178  0.85859
BMI               0.0901667  0.0184139   4.897 9.75e-07 ***
DiabetesPedigree  0.9753618  0.3045128   3.203  0.00136 **
Age               0.0104799  0.0099820   1.050  0.29377
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 682.69  on 731  degrees of freedom
AIC: 700.69

Number of Fisher Scoring iterations: 5

> # AIC = 700.69, Insulin is least significant
>
> model3 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + Glucose + BloodPressure
+              + SkinThickness + BMI + DiabetesPedigree
+              + Age)
> summary(model3)

Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
    SkinThickness + BMI + DiabetesPedigree + Age, family = binomial,
```

```
    data = diabetes)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -9.251884   0.843429 -10.969  < 2e-16 ***
Pregnancies       0.129582   0.033955   3.816 0.000135 ***
Glucose           0.037115   0.003698  10.035  < 2e-16 ***
BloodPressure    -0.007281   0.008738  -0.833 0.404735
SkinThickness     0.005311   0.013764   0.386 0.699605
BMI               0.090445   0.018356   4.927 8.34e-07 ***
DiabetesPedigree  0.977061   0.304387   3.210 0.001328 **
Age               0.010476   0.009985   1.049 0.294078
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 682.72  on 732  degrees of freedom
AIC: 698.72

Number of Fisher Scoring iterations: 5

> # AIC = 698.72, SkinThickness is least significant
>
> model4 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + Glucose + BloodPressure
+              + BMI + DiabetesPedigree + Age)
> summary(model4)

Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
    BMI + DiabetesPedigree + Age, family = binomial, data = diabetes)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -9.225133   0.839936 -10.983  < 2e-16 ***
Pregnancies       0.130061   0.033966   3.829 0.000129 ***
Glucose           0.037135   0.003698  10.043  < 2e-16 ***
BloodPressure    -0.007255   0.008740  -0.830 0.406467
BMI               0.094066   0.015823   5.945 2.76e-09 ***
DiabetesPedigree  0.977985   0.304173   3.215 0.001303 **
Age               0.010623   0.009985   1.064 0.287359
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 682.87  on 733  degrees of freedom
AIC: 696.87

Number of Fisher Scoring iterations: 5

> # AIC = 696.87, BloodPressure is least significant
>
> model5 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + Glucose + BMI
+              + DiabetesPedigree + Age)
> summary(model5)

Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigree +
    Age, family = binomial, data = diabetes)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -9.527589   0.762296 -12.499  < 2e-16 ***
Pregnancies       0.128522   0.033892   3.792 0.000149 ***
Glucose           0.036812   0.003665  10.045  < 2e-16 ***
BMI               0.090676   0.015263   5.941 2.84e-09 ***
DiabetesPedigree  0.987878   0.303293   3.257 0.001125 **
Age               0.008495   0.009639   0.881 0.378173
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 683.56  on 734  degrees of freedom
AIC: 695.56

Number of Fisher Scoring iterations: 5

> # AIC = 695.56, Age is least significant
>
> model6 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + Glucose + BMI
+              + DiabetesPedigree)
> summary(model6)

Call:
```

```
glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigree,
    family = binomial, data = diabetes)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -9.360006   0.734893 -12.737  < 2e-16 ***
Pregnancies       0.144537   0.028734   5.030 4.90e-07 ***
Glucose           0.037458   0.003604  10.395  < 2e-16 ***
BMI               0.089874   0.015256   5.891 3.84e-09 ***
DiabetesPedigree  0.998184   0.302716   3.297 0.000976 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 684.33  on 735  degrees of freedom
AIC: 694.33

Number of Fisher Scoring iterations: 5

> # AIC = 694.33, all predictors have p-values under 0.001
> # removing any further variables increases AIC
>
> # the following function takes samples from the dataset, and repeats
> # backward stepwise regressions on them, outputting what % of times
> # each predictor gets included in the model
> install.packages("bootStepAIC")
> library(bootStepAIC)
> # try 50 samples
> boot = boot.stepAIC(model2, data = diabetes, B = 50)
> boot

Summary of Bootstrapping the 'stepAIC()' procedure for

Call:
glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
    SkinThickness + Insulin + BMI + DiabetesPedigree + Age, family = binomial,
    data = diabetes)

Bootstrap samples: 50
Direction: backward
Penalty: 2 * df

Covariates selected
                 (%)
```

```
BMI              100
Glucose          100
DiabetesPedigree  98
Pregnancies       98
Age               28
BloodPressure     26
Insulin           18
SkinThickness     18

Coefficients Sign
                  + (%) - (%)
BMI              100.00  0.00
DiabetesPedigree 100.00  0.00
Glucose          100.00  0.00
Pregnancies      100.00  0.00
Age               92.86  7.14
SkinThickness     77.78 22.22
Insulin           33.33 66.67
BloodPressure      7.69 92.31

Stat Significance
                    (%)
BMI              100.00
Glucose          100.00
Pregnancies       97.96
DiabetesPedigree  89.80
BloodPressure     69.23
Age               42.86
Insulin           22.22
SkinThickness     22.22


The stepAIC() for the original data-set gave

Call:  glm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigree,
    family = binomial, data = diabetes)

Coefficients:
     (Intercept)       Pregnancies           Glucose               BMI
        -9.36001           0.14454           0.03746           0.08987
DiabetesPedigree
         0.99818

Degrees of Freedom: 739 Total (i.e. Null);  735 Residual
Null Deviance:            950.6
Residual Deviance: 684.3         AIC: 694.3
```

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
    Insulin + BMI + DiabetesPedigree + Age

Final Model:
Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigree


             Step Df   Deviance Resid. Df Resid. Dev      AIC
1                                    731   682.6879 700.6879
2       - Insulin  1 0.03183061       732   682.7198 698.7198
3 - SkinThickness  1 0.14894397       733   682.8687 696.8687
4 - BloodPressure  1 0.68903062       734   683.5577 695.5577
5           - Age  1 0.77056845       735   684.3283 694.3283

> # all 4 predictors were selected all 50 times, a good sign
>
> # we are yet to try transforms on Glucose, as suggested earlier
> # try a square root transform
> model7 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + sqrt(Glucose) + BMI
+              + DiabetesPedigree)
> summary(model7)

Call:
glm(formula = Outcome ~ Pregnancies + sqrt(Glucose) + BMI + DiabetesPedigree,
    family = binomial, data = diabetes)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -14.15643    1.09291 -12.953  < 2e-16 ***
Pregnancies        0.14475    0.02881   5.024 5.06e-07 ***
sqrt(Glucose)      0.85195    0.08185  10.408  < 2e-16 ***
BMI                0.08994    0.01528   5.885 3.97e-09 ***
DiabetesPedigree   1.00030    0.30307   3.301 0.000965 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 683.30  on 735  degrees of freedom
```

```
AIC: 693.3

Number of Fisher Scoring iterations: 5

> # AIC = 693.3, an improvement
>
> # try a log transform
> model8 = glm(data = diabetes, family = binomial,
+              formula = Outcome ~ Pregnancies + log(Glucose) + BMI
+              + DiabetesPedigree)
> summary(model8)

Call:
glm(formula = Outcome ~ Pregnancies + log(Glucose) + BMI + DiabetesPedigree,
    family = binomial, data = diabetes)

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)      -27.69039    2.32203 -11.925  < 2e-16 ***
Pregnancies        0.14499    0.02886   5.023 5.08e-07 ***
log(Glucose)       4.78478    0.46248  10.346  < 2e-16 ***
BMI                0.09010    0.01530   5.890 3.87e-09 ***
DiabetesPedigree   1.00248    0.30314   3.307 0.000943 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 950.58  on 739  degrees of freedom
Residual deviance: 683.06  on 735  degrees of freedom
AIC: 693.06

Number of Fisher Scoring iterations: 5

> # AIC = 693.06, an improvement
> # this is our final model
>
>
> # load the ToPredict.csv file
> ToPredict = read.csv("ToPredict.csv")
> head(ToPredict)
  Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
1           4     136            70             0       0 31.2
2           1     121            78            39      74 39.0
3           3     108            62            24       0 26.0
4           0     181            88            44     510 43.3
```

```
5           8     154            78            32       0 32.4
  DiabetesPedigree Age
1            1.182  22
2            0.261  28
3            0.223  25
4            0.222  26
5            0.443  45
>
> # make predictions using our model
> predict(model8, ToPredict, type = 'response')
        1          2          3          4          5
0.59667830 0.30512768 0.09226721 0.78704600 0.71781267
```