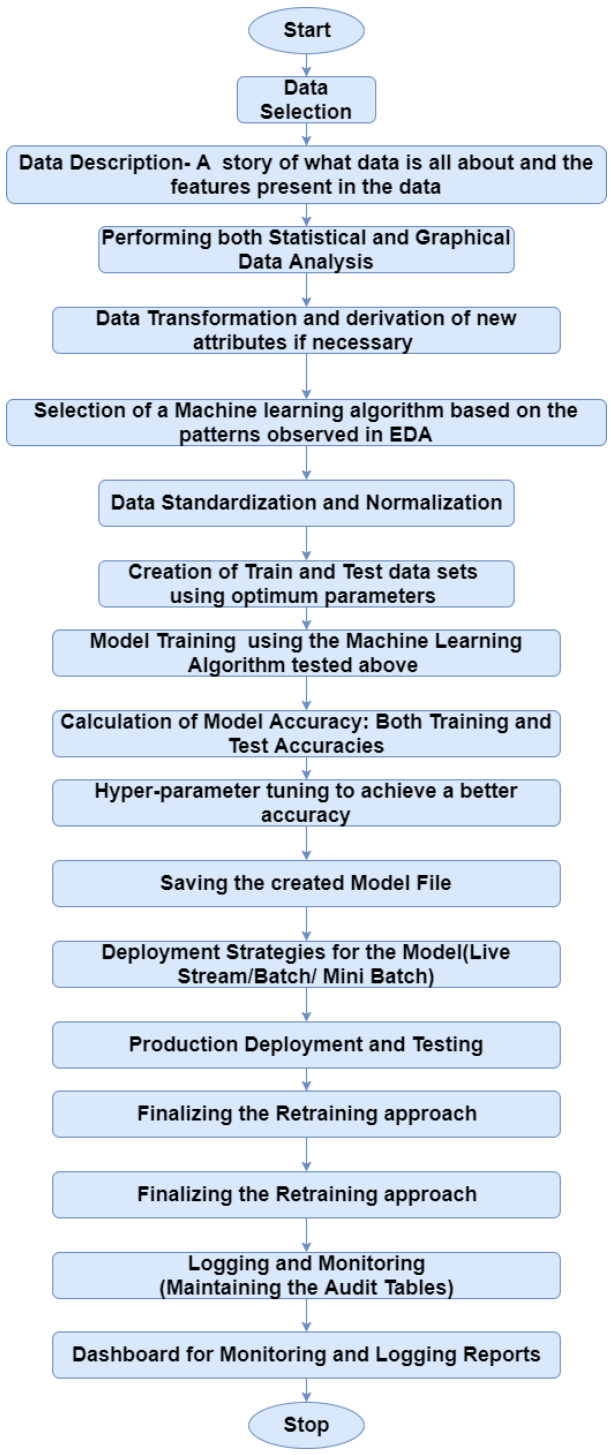
Application Flow

Logistic Regression is one of the most fundamental algorithms for classification in the Machine Learning world.

But before proceeding with the algorithm, let's first discuss the lifecycle of any machine learning model. This diagram explains the creation of a Machine Learning model from scratch and then taking the same model further with hyperparameter tuning to increase its accuracy, deciding the deployment strategies for that model and once deployed setting up the logging and monitoring frameworks to generate reports and dashboards based on the client requirements. A typical lifecycle diagram for a machine learning model looks like:



Introduction

In linear regression, the type of data we deal with is quantitative, whereas we use classification models to deal with qualitative data or categorical data. The algorithms used for solving a classification problem first predict the probability of each of the categories of the qualitative variables, as the basis for making the classification. And, as the probabilities are continuous numbers, classification using probabilities also behave like regression methods. Logistic regression is one such type of classification model which is used to classify the dependent variable into two or more classes or categories.

Why don't we use Linear regression for classification problems?

Let's suppose you took a survey and noted the response of each person as satisfied, neutral or Not satisfied. Let's map each category:

Satisfied – 2

Neutral – 1

Not Satisfied – 0

But this doesn't mean that the gap between Not satisfied and Neutral is same as Neutral and satisfied. There is no mathematical significance of these mapping. We can also map the categories like:

Satisfied – 0

Neutral – 1

Not Satisfied – 2

It's completely fine to choose the above mapping. If we apply linear regression to both the type of mappings, we will get different sets of predictions. Also, we can get prediction values like 1.2, 0.8, 2.3 etc. which makes no sense for categorical values. So, there is no normal method to convert qualitative data into quantitative data for use in linear regression. Although, for binary classification, i.e. when there only two categorical values,

using the least square method can give decent results. Suppose we have two categories Black and White and we map them as follows:

Black – 0

White - 1

We can assign predicted values for both the categories such as Y> 0.5 goes to class white and vice versa. Although, there will be some predictions for which the value can be greater than 1 or less than 0 making them hard to classify in any class. Nevertheless, linear regression can work decently for binary classification but not that well for multi-class classification. Hence, we use classification methods for dealing with such problems.

Logistic Regression

Logistic regression is one such regression algorithm which can be used for performing classification problems. It calculates the probability that a given value belongs to a specific class. If the probability is more than 50%, it assigns the value in that particular class else if the probability is less than 50%, the value is assigned to the other class. Therefore, we can say that logistic regression acts as a binary classifier.

Working of a Logistic Model

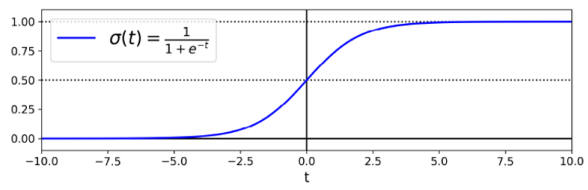
For linear regression, the model is defined by: $y = \beta_0 + \beta_1 x$ - (i)

and for logistic regression, we calculate probability, i.e. y is the probability of a given variable x belonging to a certain class. Thus, it is obvious that the value of y should lie between 0 and 1.

But, when we use equation(i) to calculate probability, we would get values less than 0 as well as greater than 1. That doesn't make any sense . So, we need to use such an equation which always gives values between 0 and 1, as we desire while calculating the probability.

Sigmoid function

We use the sigmoid function as the underlying function in Logistic regression. Mathematically and graphically, it is shown as:



Why do we use the Sigmoid Function?

1) The sigmoid function's range is bounded between 0 and 1. Thus it's useful in calculating the probability for the Logistic function. 2) It's derivative is easy to calculate than other functions which is useful during gradient descent calculation. 3) It is a simple way of introducing non-linearity to the model.

Although there are other functions as well, which can be used, but sigmoid is the most common function used for logistic regression.

The logistic function is given as:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

Evaluation of a Classification Model

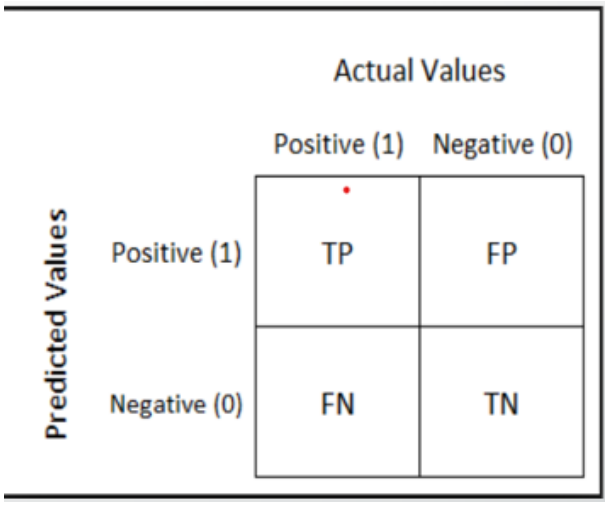
In machine learning, once we have a result of the classification problem, how do we measure how accurate our classification is? For a regression problem, we have different metrics like R Squared score, Mean Squared Error etc. what are the metrics to measure the credibility of a classification model?

Metrics In a regression problem, the accuracy is generally measured in terms of the difference in the actual values and the predicted values. In a classification problem, the credibility of the model is measured using the confusion matrix generated, i.e., how accurately the true positives and true negatives were predicted. The different metrics used for this purpose are:

- Accuracy
- Recall
- Precision
- F1 Score
- Specifity
- AUC(Area Under the Curve)
- ROC(Receiver Operator Characteristic)

Confusion Matrix

A typical confusion matrix looks like the figure shown.



Where the terms have the meaning:

 True Positive(TP): A result that was predicted as positive by the classification model and also is positive

 True Negative(TN): A result that was predicted as negative by the classification model and also is negative

 False Positive(FP): A result that was predicted as positive by the classification model but actually is negative

 False Negative(FN): A result that was predicted as negative by the classification model but actually is positive.

The Credibility of the model is based on how many correct predictions did the model do.

Accuracy

The mathematical formula is :

Accuracy= $\frac{(TP+TN)}{(TP+TN+FP+FN)}$

Or, it can be said that it's defined as the total number of correct classifications divided by the total number of classifications.

Recall or Sensitivity

The mathematical formula is:

Recall= $\frac{TP}{(TP+FN)}$

Or, as the name suggests, it is a measure of: from the total number of positive results how many positives were correctly predicted by the model.

It shows how relevant the model is, in terms of positive results only.

Let's suppose in the previous model, the model gave 50 correct predictions(TP) but failed to identify 200 cancer patients(FN). Recall in that case will be:

Recall= $\frac{50}{(50+200)}$ = 0.2 (The model was able to recall only 20% of the cancer patients)

Precision

Precision is a measure of amongst all the positive predictions, how many of them were actually positive. Mathematically,

Precision= $\frac{TP}{(TP+FP)}$

Let's suppose in the previous example, the model identified 50 people as cancer patients(TP) but also raised a false alarm for 100 patients(FP). Hence,

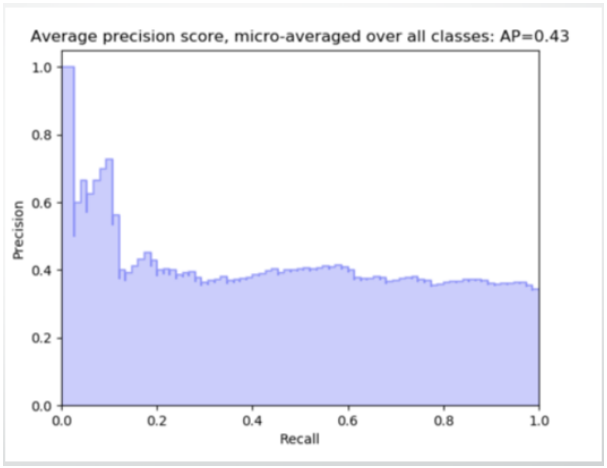
Precision= $\frac{50}{(50+100)}$ =0.33 (The model only has a precision of 33%)

But we have a problem!!

As evident from the previous example, the model had a very high Accuracy but performed poorly in terms of Precision and Recall. So, necessarily *Accuracy* is not the metric to use for evaluating the model in this case.

Imagine a scenario, where the requirement was that the model recalled all the defaulters who did not pay back the loan. Suppose there were 10 such defaulters and to recall those 10 defaulters, and the model gave you 20 results out of which only the 10 are the actual defaulters. Now, the recall of the model is 100%, but the precision goes down to 50%.

A Trade-off?



As observed from the graph, with an increase in the Recall, there is a drop in Precision of the model.

So the question is - what to go for? Precision or Recall?

Well, the answer is: it depends on the business requirement.

For example, if you are predicting cancer, you need a 100 % recall. But suppose you are predicting whether a person is innocent or not, you need 100% precision.

Can we maximise both at the same time? No

So, there is a need for a better metric then?

Yes. And it's called an *F1 Score*

F1 Score

From the previous examples, it is clear that we need a metric that considers both Precision and Recall for evaluating a model. One such metric is the F1 score.

F1 score is defined as the harmonic mean of Precision and Recall.

The mathematical formula is: F1 score= $\frac{2*((Precision*Recall)}{(Precision+Recall))}$

Specificity or True Negative Rate

This represents how specific is the model while predicting the True Negatives. Mathematically,

Specificity=$\frac{TN}{(TN+FP)}$ Or, it can be said that it quantifies the total number of negatives predicted by the model with respect to the total number of actual negative or non favorable outcomes.

Similarly, False Positive rate can be defined as: (1- specificity) Or, $\frac{FP}{(TN+FP)}$

ROC(Receiver Operator Characteristic)

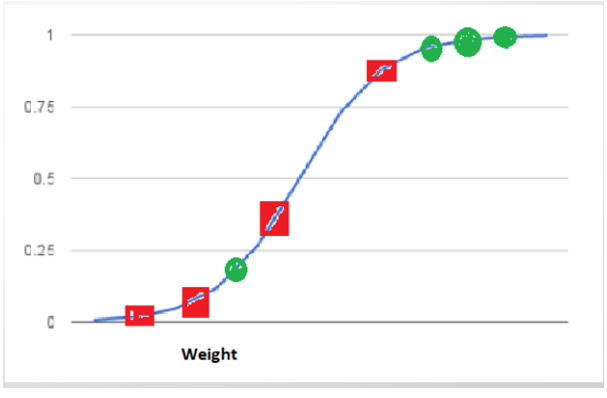
We know that the classification algorithms work on the concept of probability of occurrence of the possible outcomes. A probability value lies between 0 and 1. Zero means that there is no probability of occurrence and one means that the occurrence is certain.

But while working with real-time data, it has been observed that we seldom get a perfect 0 or 1 value. Instead of that, we get different decimal values lying between 0 and 1. Now the question is if we are not getting binary probability values how are we actually determining the class in our classification problem?

There comes the concept of Threshold. A threshold is set, any probability value below the threshold is a negative outcome, and anything more than the threshold is a favourable or the positive outcome. For Example, if the threshold is 0.5, any probability value below 0.5 means a negative or an unfavourable outcome and any value above 0.5 indicates a positive or favourable outcome.

Now, the question is, what should be an ideal threshold?

The following diagram shows a typical logistic regression curve.

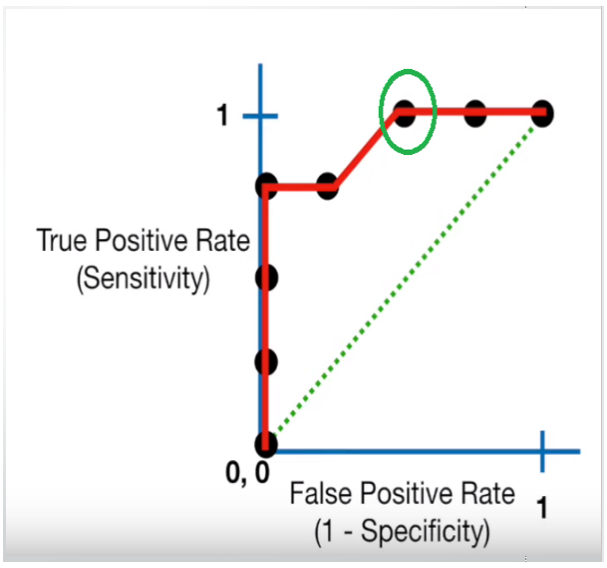


- The horizontal lines represent the various values of thresholds ranging from 0 to 1.
- Let's suppose our classification problem was to identify the obese people from the given data.
- The green markers represent obese people and the red markers represent the non-obese people.
- Our confusion matrix will depend on the value of the threshold chosen by us.

- For Example, if 0.25 is the threshold then

```
TP(actually obese)=3
TN(Not obese)=2
FP(Not obese but predicted obese)=2(the two red squares above the 0.25 line)
FN(Obese but predicted as not obese )=1(Green circle below 0.25line  )
```

A typical ROC curve looks like the following figure.

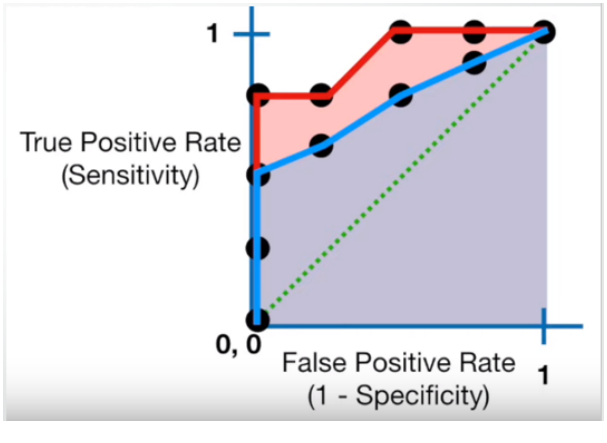


- Mathematically, it represents the various confusion matrices for various thresholds. Each black dot is one confusion matrix.
- The green dotted line represents the scenario when the true positive rate equals the false positive rate.
- As evident from the curve, as we move from the rightmost dot towards left, after a certain threshold, the false positive rate decreases.
- After some time, the false positive rate becomes zero.
- The point encircled in green is the best point as it predicts all the values correctly and keeps the False positive as a minimum.
- But that is not a rule of thumb. Based on the requirement, we need to select the point of a threshold.
- The ROC curve answers our question of which threshold to choose.

But we have a confusion!!

Let’s suppose that we used different classification algorithms, and different ROCs for the corresponding algorithms have been plotted. The question is: which algorithm to choose now? The answer is to calculate the area under each ROC curve.

AUC(Area Under Curve)



- It helps us to choose the best model amongst the models for which we have plotted the ROC curves
- The best model is the one which encompasses the maximum area under it.
- In the adjacent diagram, amongst the two curves, the model that resulted in the red one should be chosen as it clearly covers more area than the blue one

Python Implementation

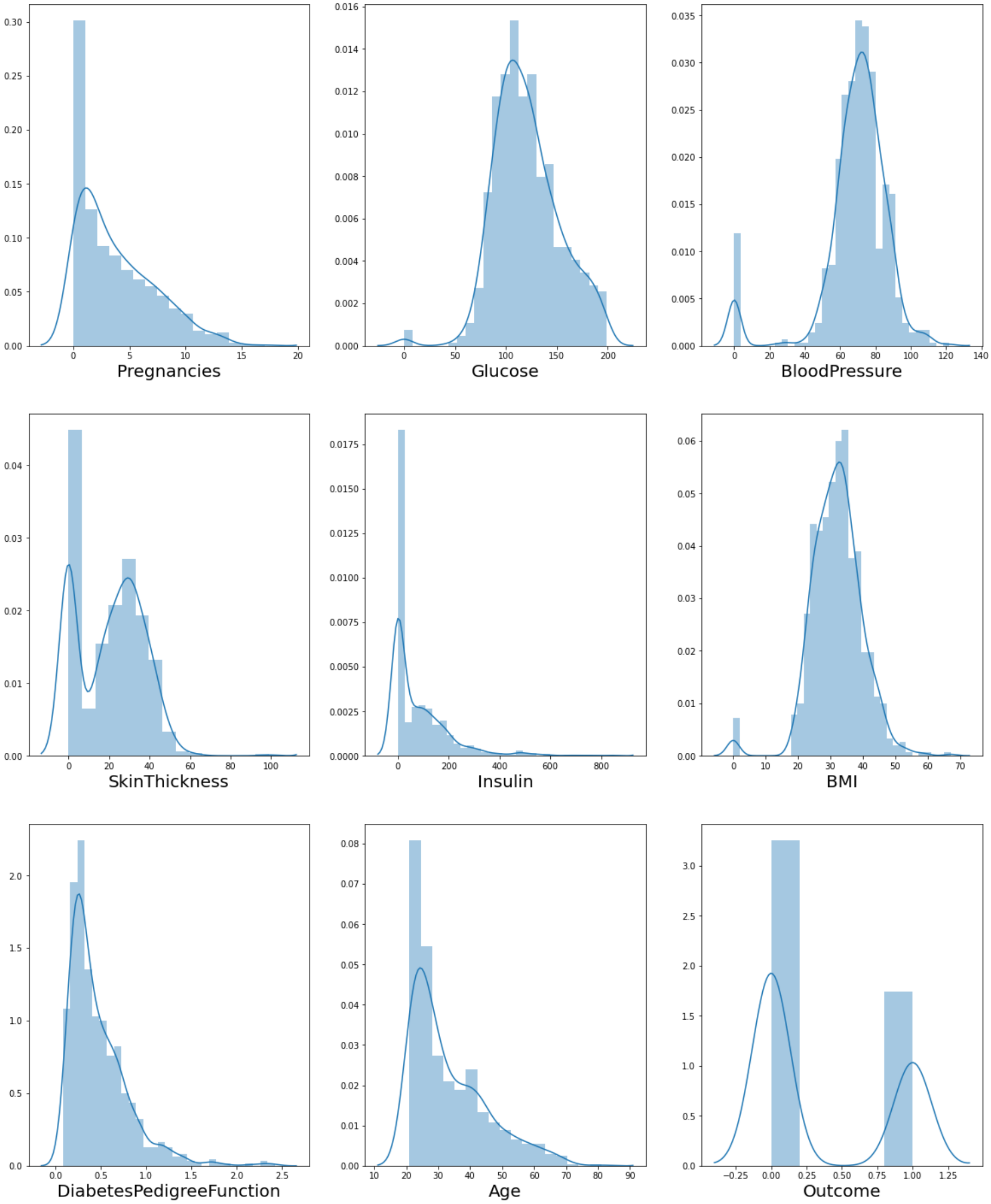
For more documentation visit,

scikit-learn.org

In [ ]:
```
# Refer the live class video for codes
```

It seems that there are no missing values in our data. Great, let's see the distribution of data:

In [4]:
```
# Refer the live class video for codes
```

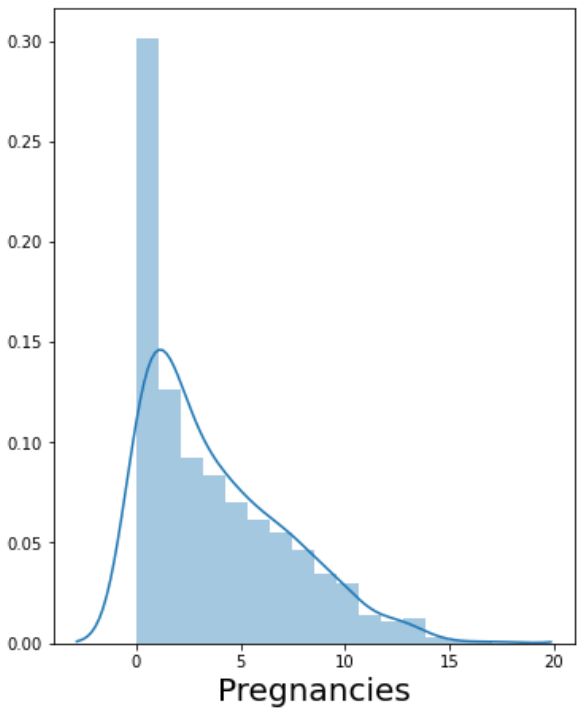


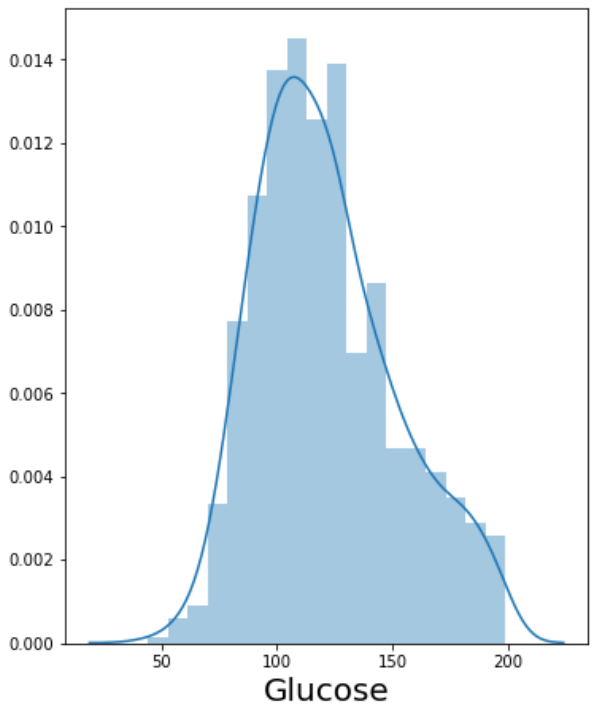
We can see there is some skewness in the data, let's deal with data.

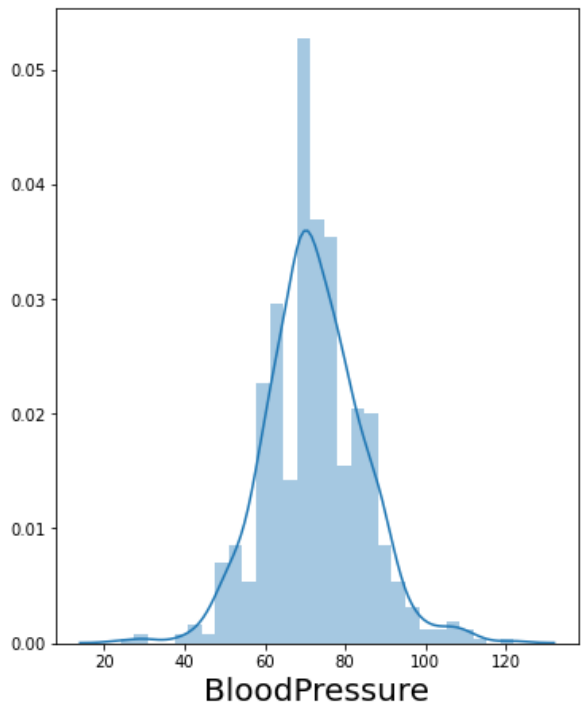
Also, we can see there few data for columns Glucose, Insulin, skin thickness, BMI and Blood Pressure which have value as 0. That's not possible. You can do a quick search to see that one cannot have 0 values for these. Let's deal with that. we can either remove such data or simply replace it with their respective mean values. Let's do the latter.

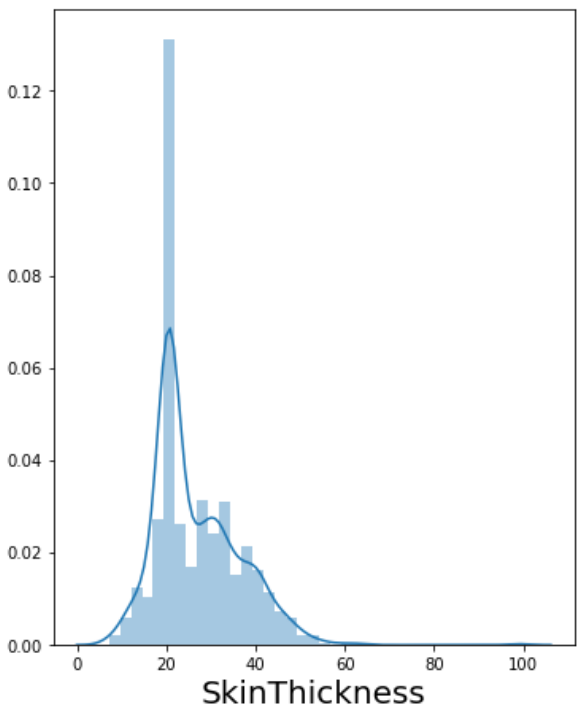
In [5]:
```
# Refer the live class video for codes
```

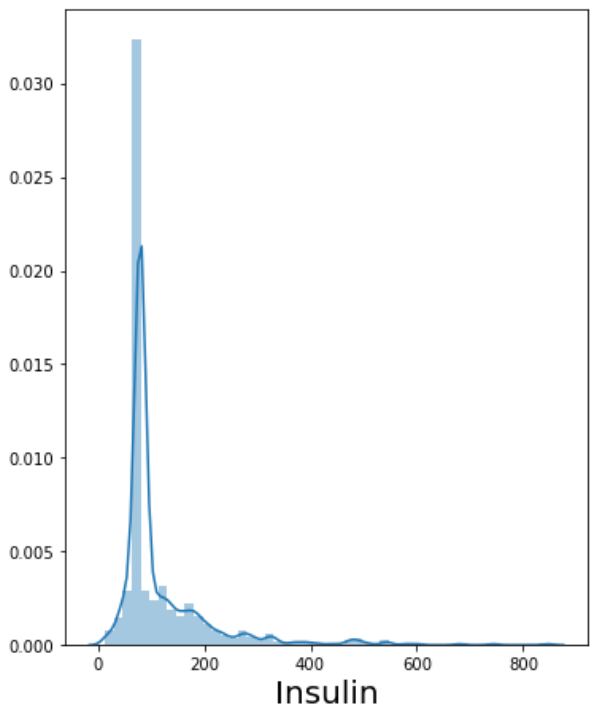
In [6]:
```
# Refer the live class video for codes
```

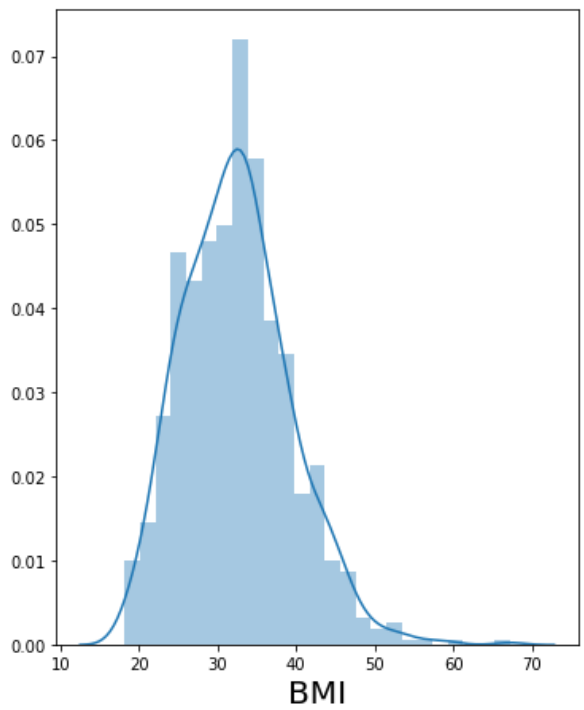


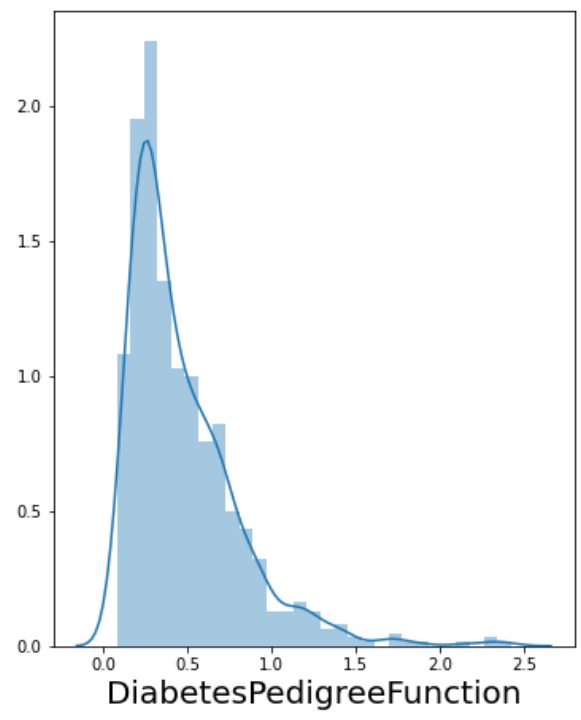


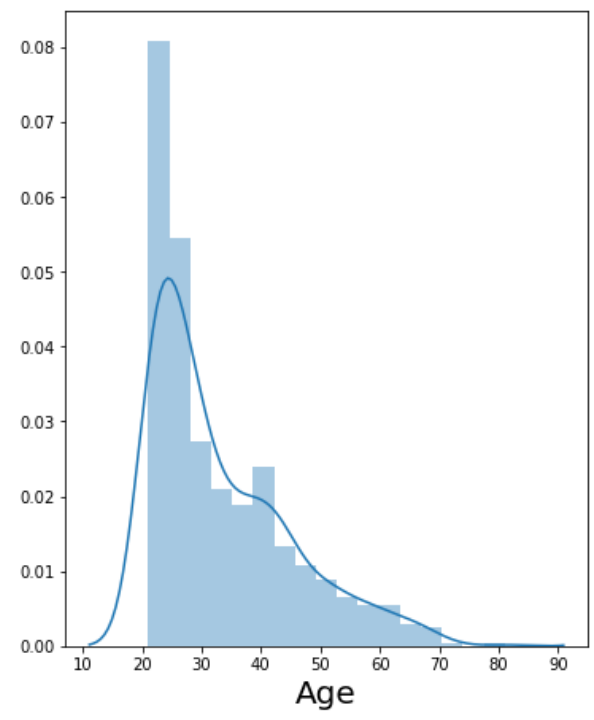


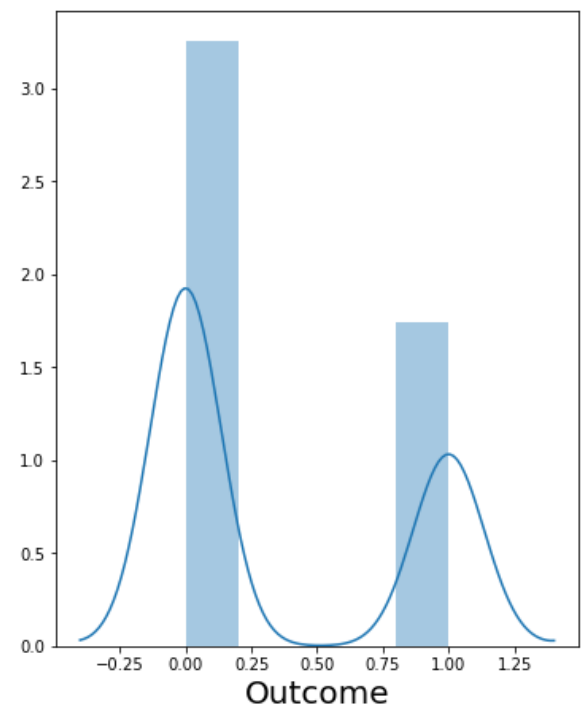








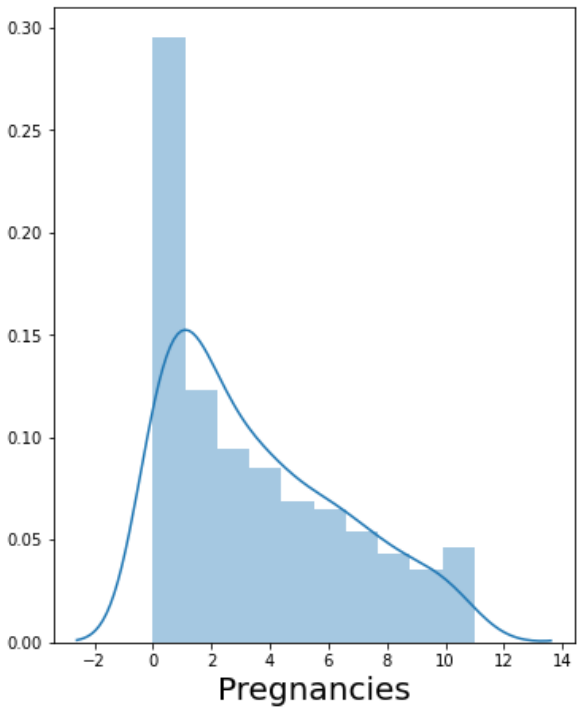


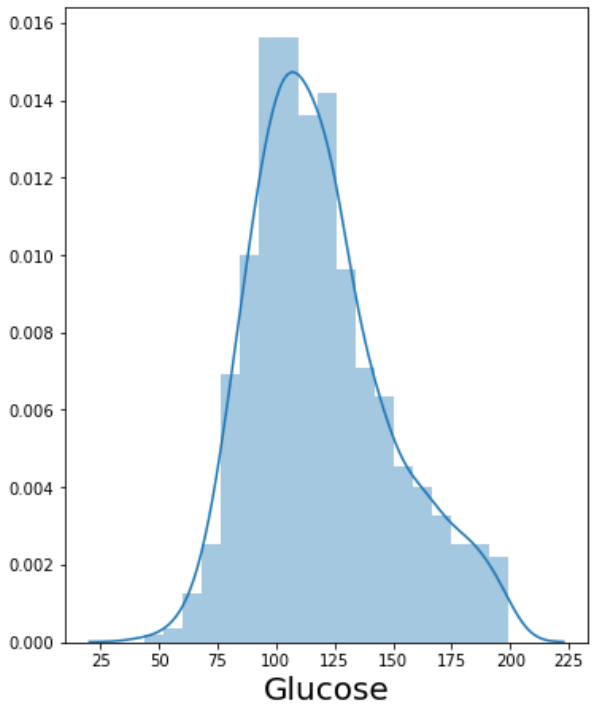


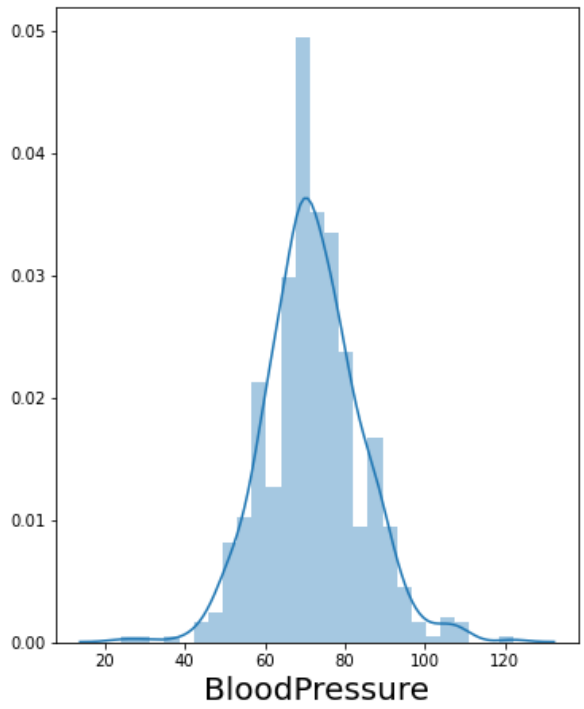
Now we have dealt with the 0 values and data looks better. But, there still are outliers present in some columns. Let's deal with them.

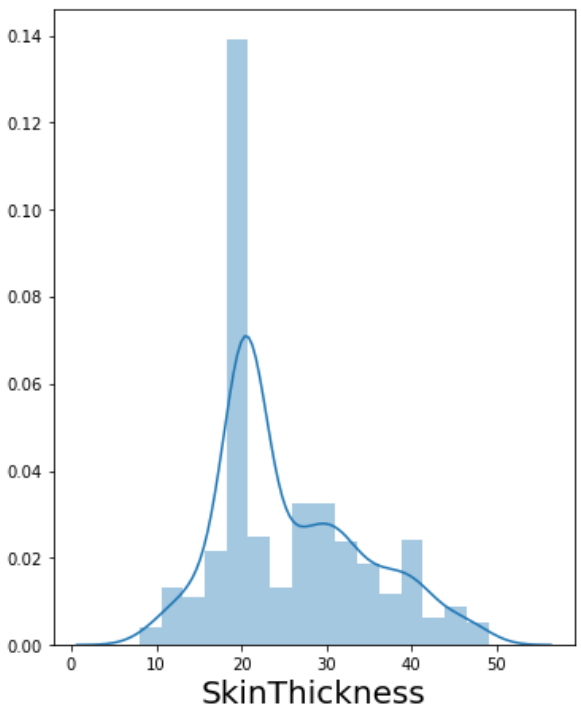
In [ ]:
```
# Refer the live class video for codes
```

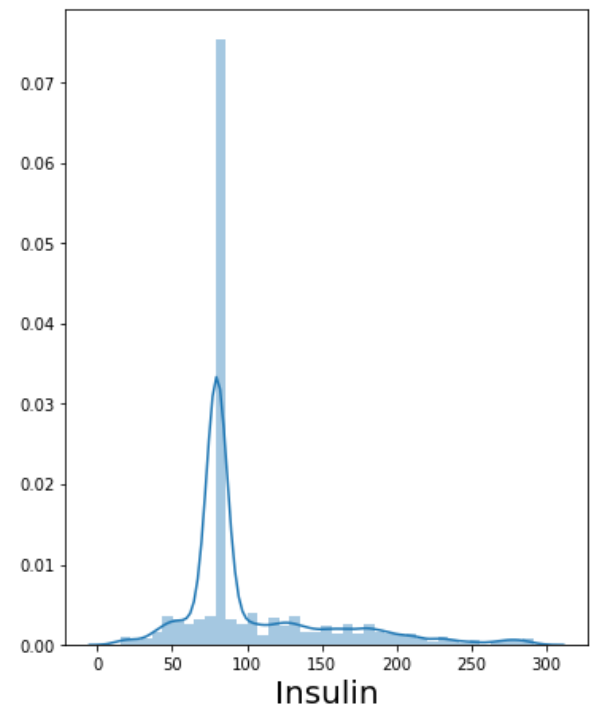
In [8]:
```
# Refer the live class video for codes
```

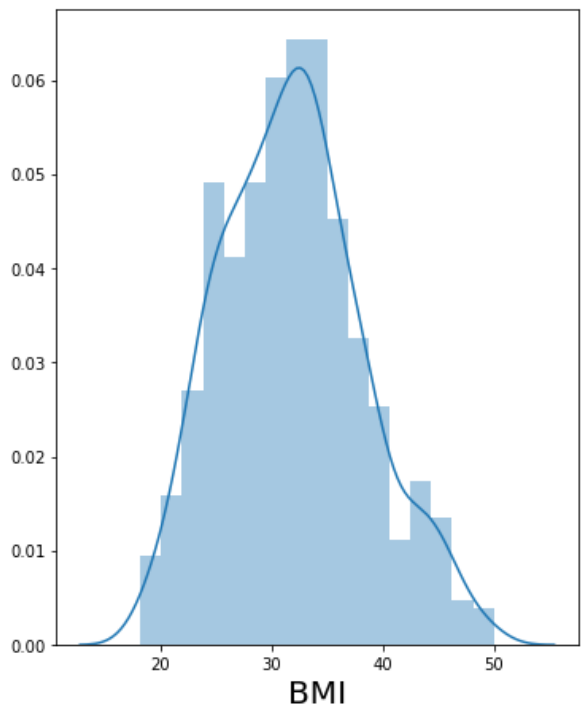


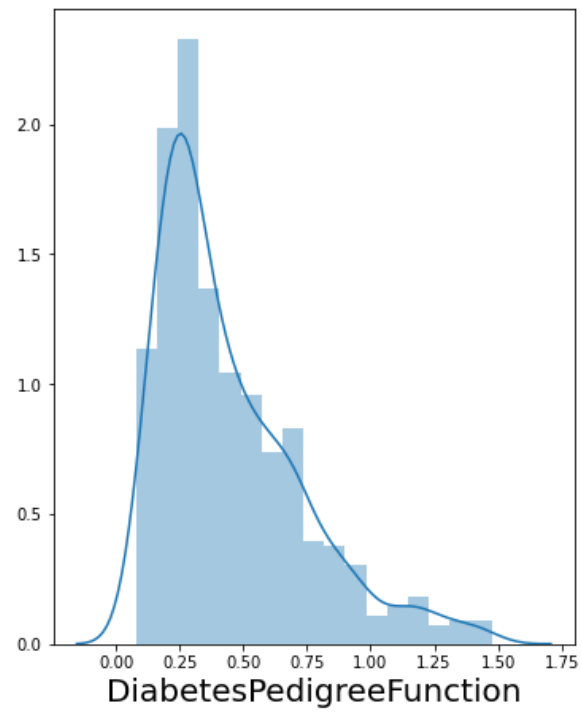


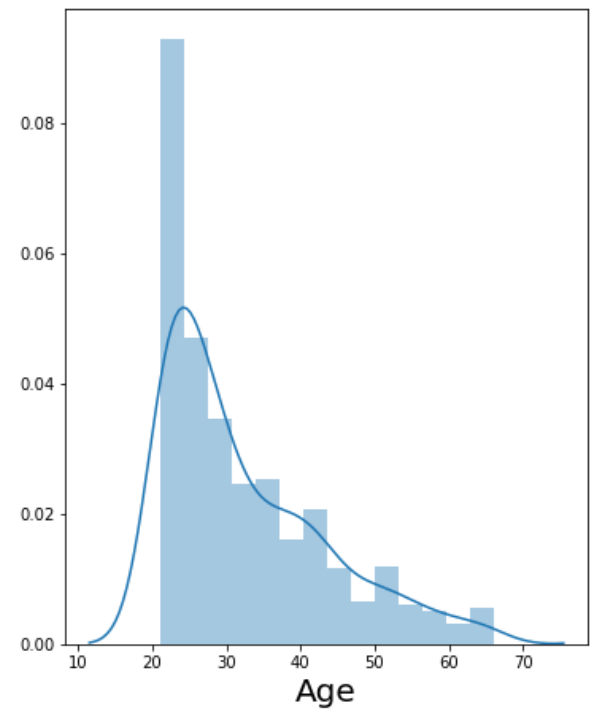


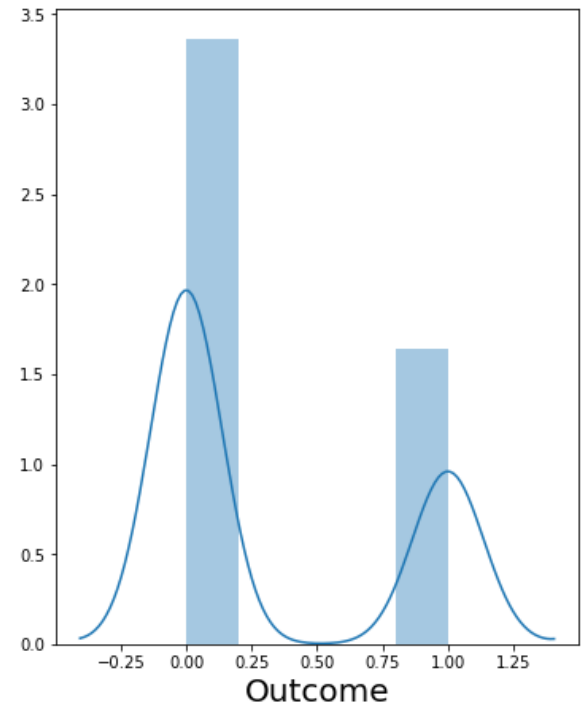










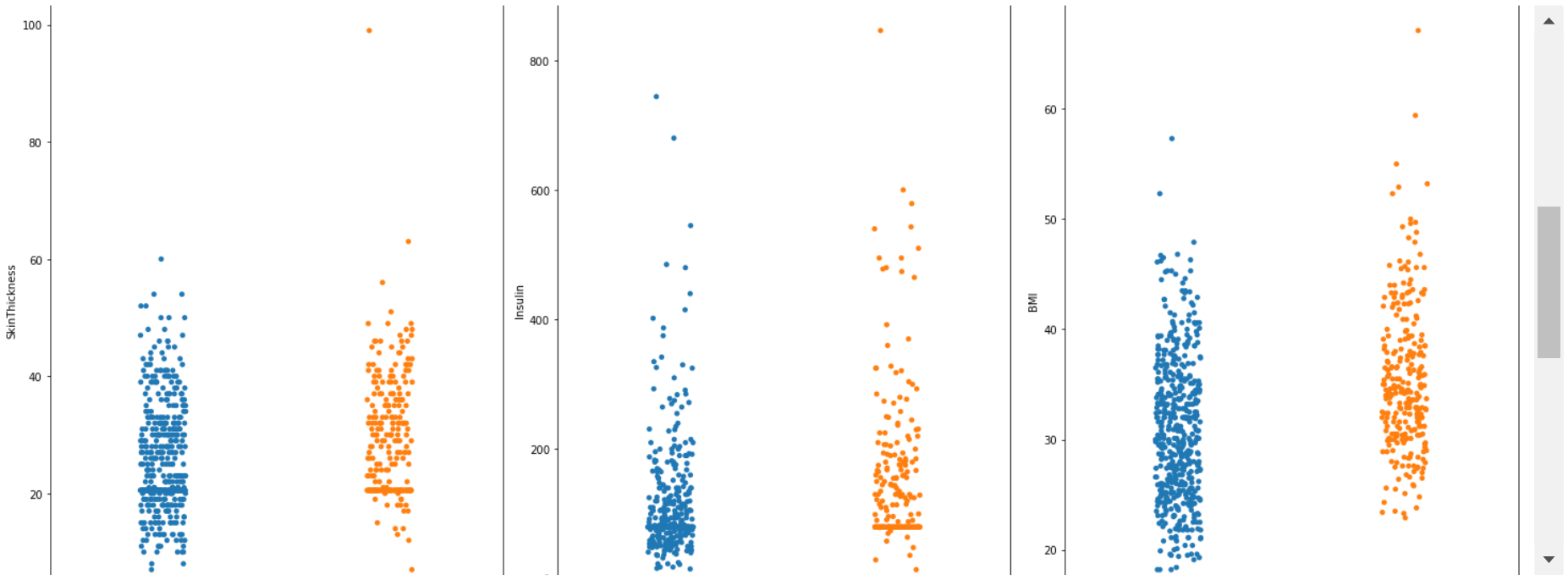


The data looks much better now than before. We will start our analysis with this data now as we don't want to lose important information. If our model doesn't work with accuracy, we will come back for more preprocessing.

In [9]:
```
# Refer the live class video for codes
```

Before we fit our data to a model, let's visualize the relationship between our independent variables and the categories.

In [10]:
```
# Refer the live class video for codes
```



Great!! Let's proceed by checking multicollinearity in the dependent variables. Before that, we should scale our data. Let's use the standard scaler for that.

In [12]:
```
# Refer the live class video for codes
```

This is how our data looks now after scaling. Great, now we will check for multicollinearity using VIF(Variance Inflation factor)

In [ ]:
```
# Refer the live class video for codes
```

In [13]:
```
# Refer the live class video for codes
```

Out[13]:

	vif	Features
0	1.431075	Pregnancies
1	1.347308	Glucose
2	1.247914	BloodPressure
3	1.450510	SkinThickness
4	1.262111	Insulin
5	1.550227	BMI
6	1.058104	DiabetesPedigreeFunction
7	1.605441	Age

All the VIF values are less than 5 and are very low. That means no multicollinearity. Now, we can go ahead with fitting our data to the model. Before that, let's split our data in test and training set.

In [ ]:
```
# Refer the live class video for codes
```

In [ ]:
```
# Refer the live class video for codes
```

Let's see how well our model performs on the test data set.

In [ ]:
```
# Refer the live class video for codes
```

ROC

#fpr -False Positive Rate increasing frequency #tpr -True Positive Rate increasing frequency #thresholds -Decreasing thresholds on the decision function used to compute fpr and tpr. fpr, tpr, thresholds = roc_curve(y_test, y_pred)

In [ ]:
```
# Refer the live class video for codes
```

What is the significance of Roc curve and AUC?

In real life, we create various models using different algorithms that we can use for classification purpose. We use AUC to determine which model is the best one to use for a given dataset. Suppose we have created Logistic regression, SVM as well as a clustering model for classification purpose. We will calculate AUC for all the models seperately. The model with highest AUC value will be the best model to use.

Advantages of Logisitic Regression

- It is very simple and easy to implement.
- The output is more informative than other classification algorithms
- It expresses the relationship between independent and dependent variables
- Very effective with linearly seperable data

Disadvantages of Logisitic Regression

- Not effective with data which are not linearly seperable
- Not as powerful as other classification models
- Multiclass classifications are much easier to do with other algorithms than logisitic regression
- It can only predict categorical outcomes

In [ ]:
```
1
```