MACHINE LEARNING ASSIGNMENT

DOCUMENTATION

MODEL BUILDING FOR PIMA INDIANS DIABETES

Team Members :

I. Cherish (IMT2017022)
S. Sanjay (IMT2017037)
S. Prasanth (IMT2017525)

DATA PREPROCESSING :

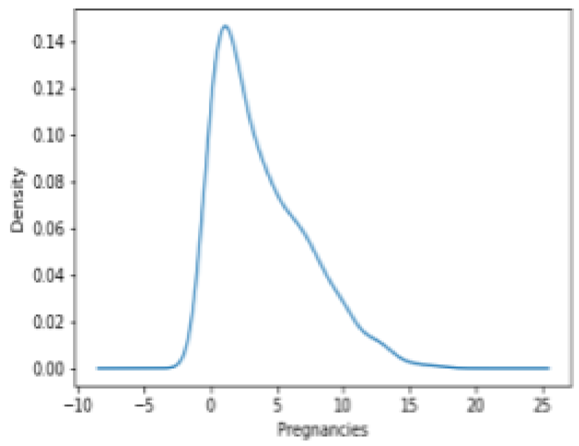
The given dataset contains invalid data, like negative values and zeros. It also has missing values, which needs to be filled in with valid data. As part of data preprocessing, all negative values have been converted to zeros. Next, the zeros of all features, except Pregnancies and Outcome(since both the features can be zeros), have been converted to NaNs. These NaNs are filled with data in the next part.

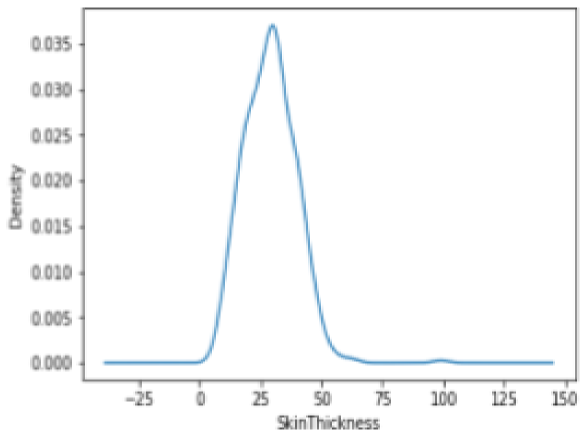
Some records might have too NaNs in them. These records might not provide much information about the problem. So we removed the records that have more than 4 NaNs in them.

MISSING DATA HANDLING :

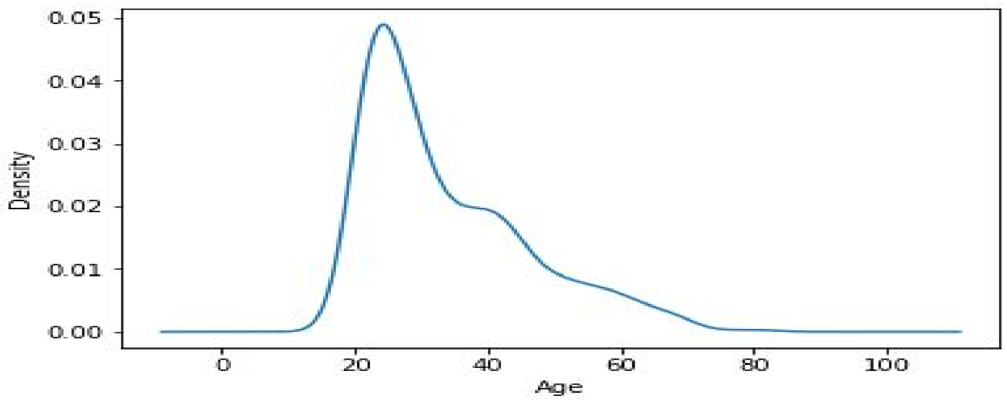
Pregnancies, Age and SkinThickness :

For Pregnancies and SkinThickness we plotted the distribution and it followed gaussian distribution. For SkinThickness, the values are accumulated between 25 and 35. Therefore, we replaced the missing values using mode because the distribution has a single long peak. For Pregnancies we used median to replace the missing data because the distribution is spread over a range and since we have removed the outliers. We have also added small noise to both the values so that the new data also follows a similar distribution.





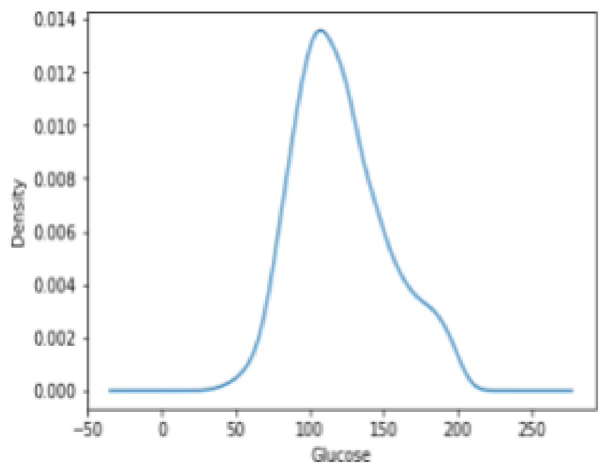
Age doesn’t depend on any other feature since it is an independent variable. We replaced the missing data in age with the mean since it has more spread across the range.

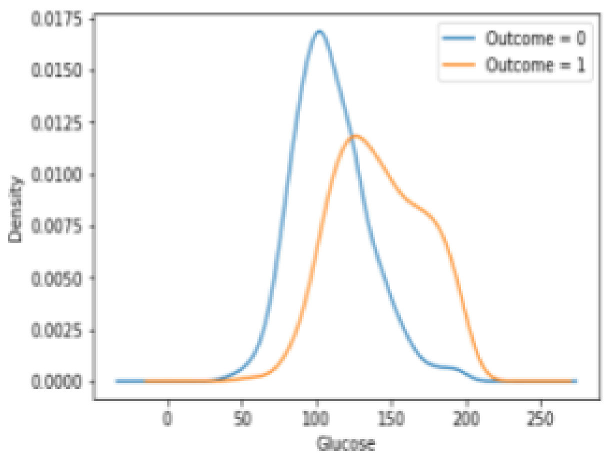


Glucose,DiabetesPedigreeFunction and Insulin :

From the correlation table it is observed that the outcome is highly correlated with glucose,diabetes pedigree function and insulin. Therefore we used grouping based on outcome. We divided the features, glucose,diabetes pedigree function, and insulin, into two groups based on the outcome feature. Then we plotted their distribution and observed that the new distribution is different compared to the distribution before grouping.

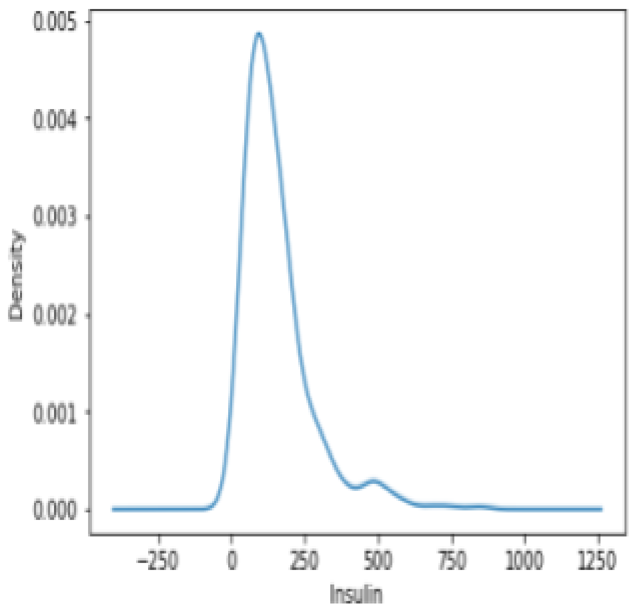
Left plot shows the distribution before grouping and the right plot shows the distribution after grouping.

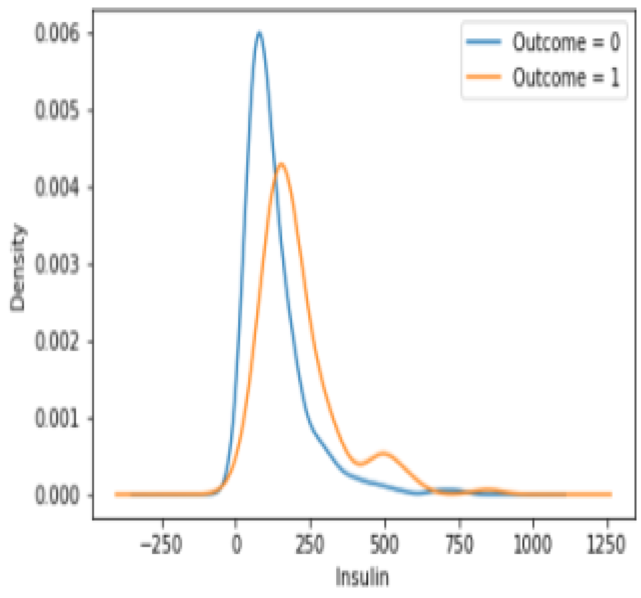


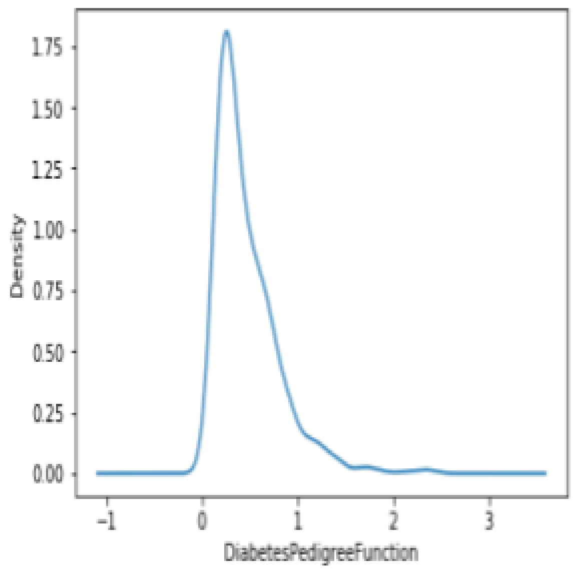


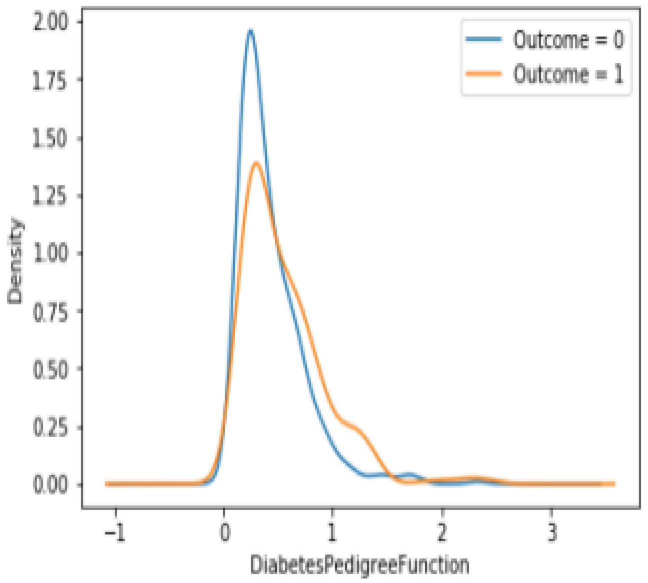
From the right plot, we can see that the group 'outcome = 0' has a single thin peak. So we can use the mode of the group to replace the missing values. On the other hand, the group 'outcome = 1' has data spreadover a large range, therefore we replaced the missing data with the mean of the group.

For insulin, from the right plot we can see that both the groups have a single long thin peak. Therefore we have replaced the missing values with the mode of their respective groups. Similarly, the feature, DiabetesPedigreeFunction also has long thin peaks. So we used the mode of their respective groups the missing values. We have also added small noise so that the new data also follows a similar distribution.



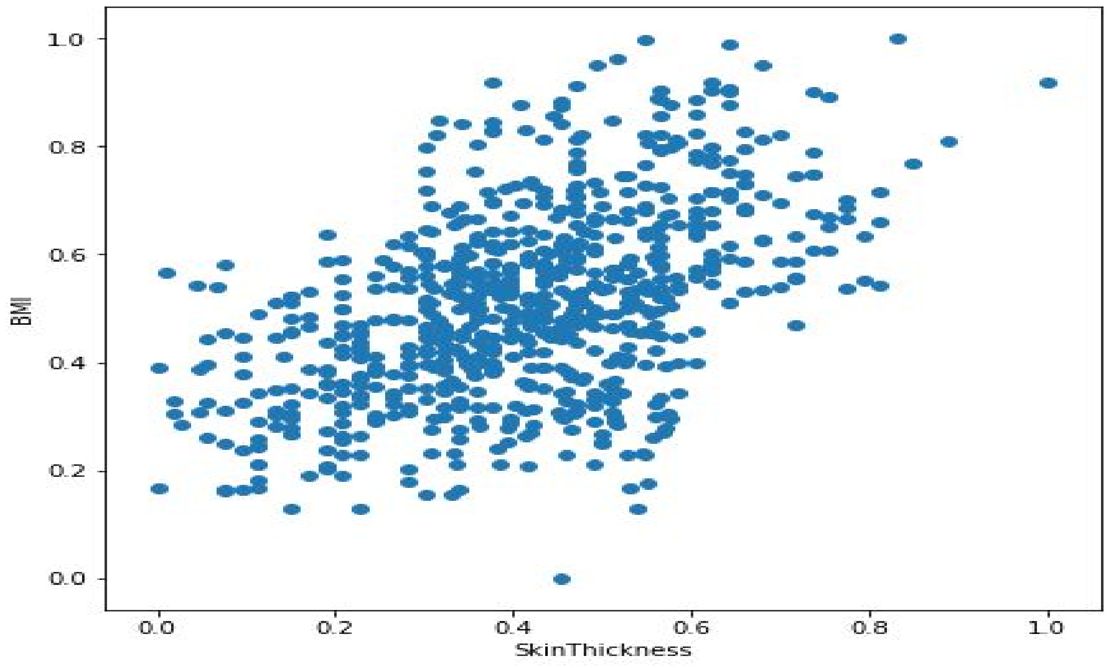






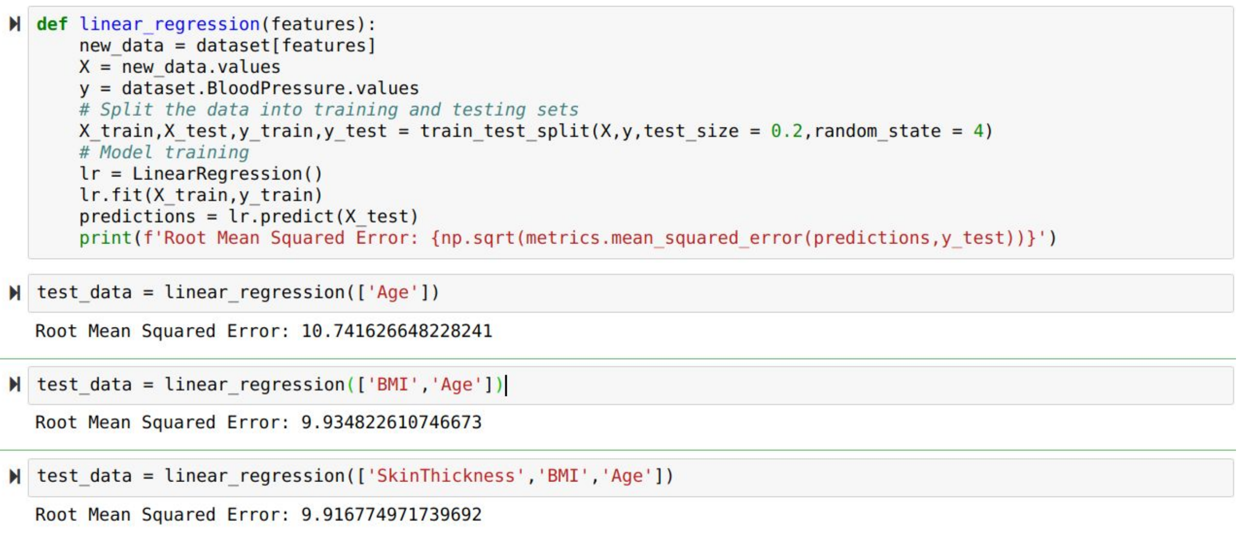
BMI :

BMI is highly correlated with SkinThickness, so we used SkinThickness to find the missing values of BMI using linear regression.



Blood Pressure :

Blood pressure is correlated with BMI, Age and SkinThickness. But when linear regression has been used with SkinThickness, the root mean square error did not decrease as expected. So, only BMI and Age have been used to predict the missing values of BloodPressure.



```
def linear_regression(features):
    new_data = dataset[features]
    X = new_data.values
    y = dataset.BloodPressure.values
    # Split the data into training and testing sets
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2,random_state = 4)
    # Model training
    lr = LinearRegression()
    lr.fit(X_train,y_train)
    predictions = lr.predict(X_test)
    print(f'Root Mean Squared Error: {np.sqrt(metrics.mean_squared_error(predictions,y_test))}')
```

```
test_data = linear_regression(['Age'])
```

```
Root Mean Squared Error: 10.741626648228241
```

```
test_data = linear_regression(['BMI','Age'])
```

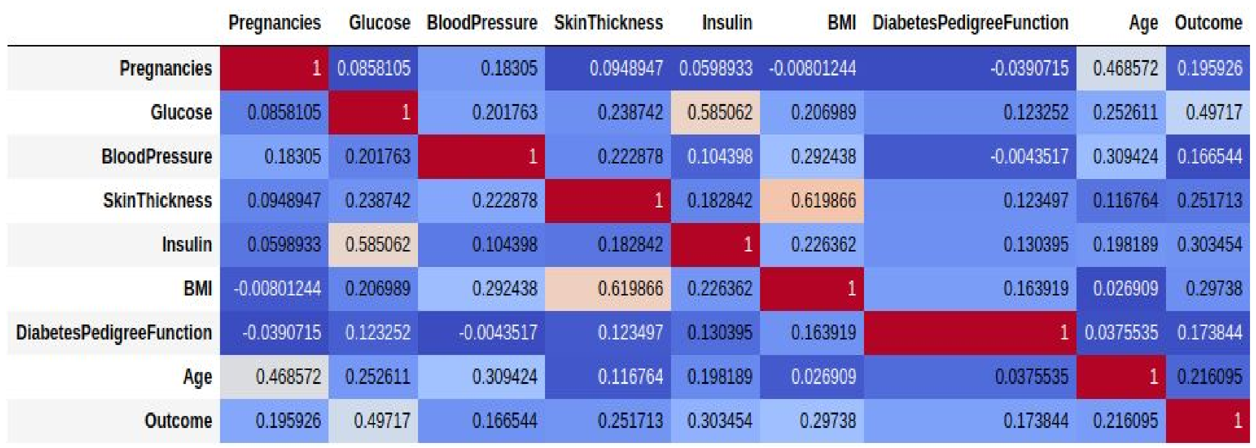
```
Root Mean Squared Error: 9.934822610746673
```

```
test_data = linear_regression(['SkinThickness','BMI','Age'])
```

```
Root Mean Squared Error: 9.916774971739692
```

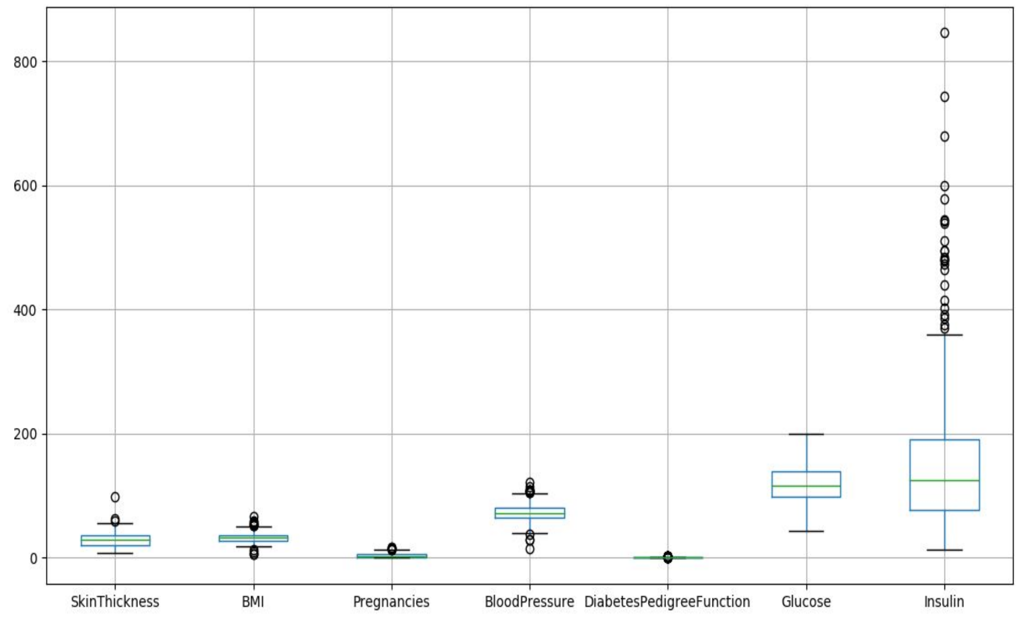
EXPLORATORY DATA ANALYSIS :

We used the correlation matrix to find how features are related with each other. We used these dependent features to fill the missing values.



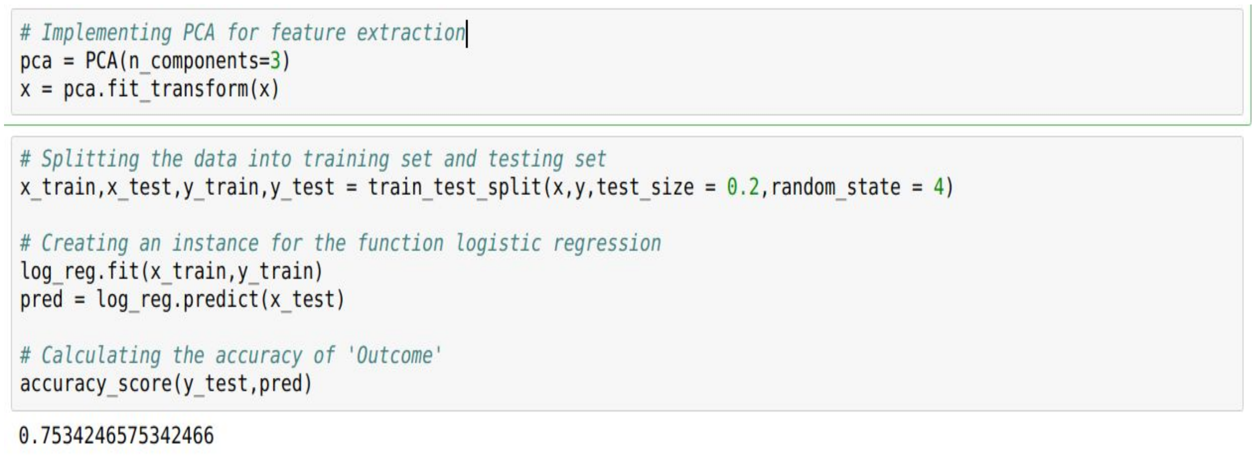
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.0858105	0.18305	0.0948947	0.0598933	-0.00801244	-0.0390715	0.468572	0.195926
Glucose	0.0858105	1	0.201763	0.238742	0.585062	0.206989	0.123252	0.252611	0.49717
BloodPressure	0.18305	0.201763	1	0.222878	0.104398	0.292438	-0.0043517	0.309424	0.166544
SkinThickness	0.0948947	0.238742	0.222878	1	0.182842	0.619866	0.123497	0.116764	0.251713
Insulin	0.0598933	0.585062	0.104398	0.182842	1	0.226362	0.130395	0.198189	0.303454
BMI	-0.00801244	0.206989	0.292438	0.619866	0.226362	1	0.163919	0.026909	0.29738
DiabetesPedigreeFunction	-0.0390715	0.123252	-0.0043517	0.123497	0.130395	0.163919	1	0.0375535	0.173844
Age	0.468572	0.252611	0.309424	0.116764	0.198189	0.026909	0.0375535	1	0.216095
Outcome	0.195926	0.49717	0.166544	0.251713	0.303454	0.29738	0.173844	0.216095	1

Outliers are the extreme cases that might affect our model and might give undesired outcomes. Therefore we used box plots to remove the outliers.



Feature Reduction :

Since the number of features in the given dataset are less, reducing the number of features will not affect the computation time of the model. Using PCA will decrease the accuracy of the model with little to no effect on computation.



```
# Implementing PCA for feature extraction
pca = PCA(n_components=3)
x = pca.fit_transform(x)
```

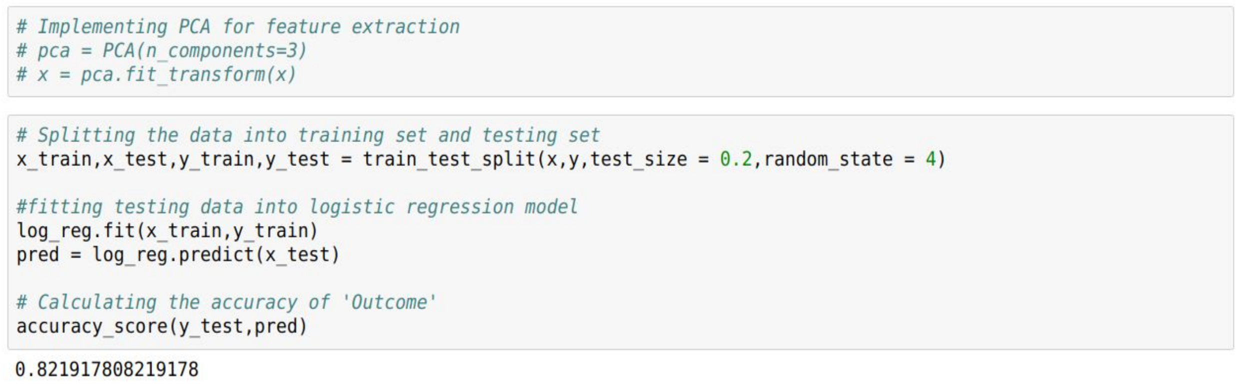
```
# Splitting the data into training set and testing set
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 4)

# Creating an instance for the function logistic regression
log_reg.fit(x_train,y_train)
pred = log_reg.predict(x_test)

# Calculating the accuracy of 'Outcome'
accuracy_score(y_test,pred)
```

```
0.7534246575342466
```

The above picture shows the accuracy that we got when we used PCA and the picture below shows the accuracy when we included all the features.



```
# Implementing PCA for feature extraction
# pca = PCA(n_components=3)
# x = pca.fit_transform(x)
```

```
# Splitting the data into training set and testing set
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 4)

#fitting testing data into logistic regression model
log_reg.fit(x_train,y_train)
pred = log_reg.predict(x_test)

# Calculating the accuracy of 'Outcome'
accuracy_score(y_test,pred)
```

```
0.821917808219178
```

MODEL BUILDING :

First, we normalize the data so that all features have the same range. The features in un-normalized data might have large differences in their ranges and it might create ill-conditioning in the analysis of our data. So normalizing the features into a single range helps in the analysis.

In the given problem the outcome is dichotomous i.e it can take only two values, either a 0 or 1. This is classification problem, since we have to find if a given person has diabetes. Logistic regression can be used to solve classification problems. We divided the given dataset into training set and testing set. We achieved an accuracy of **0.8219** using Logistic Regression**.**