

Diabetes Prediction Using Different Machine Learning Approaches

Introduction

Diabetes is one of the diseases that is spreading the fastest in the world. It can lead to a number of serious problems, such as heart disease, kidney failure, diabetic retinopathy, and neuropathy, all of which increase the number of people who get sick and die. If diabetes is caught early, its severity and the risk factors that because it can be lessened by a lot. According to the World Health Organization (WHO), diabetes causes 1.6 million deaths annually and affects 8.5% of individuals over the age of 18. (World Health Organization, 2021). Because of the high diabetes fatality and risk factor rates, as well as the potential consequences, it is critical to understand how to manage diabetes and avoid such problems. There is no question that this startling statistic requires immediate attention. Machine learning has been implemented in many sectors of medical health due to its fast growth. Machine learning and data mining methods may be used to diabetes-associated datasets to lower the likelihood of acquiring certain significant problems connected to diabetes.

Machine learning is a subfield of artificial intelligence and computer science that use data and algorithms to simulate how people learn. By developing models using patient-collected datasets, machine learning approaches improve prediction accuracy. The study is built on ML techniques such as Logistic Regression(LR), Naive Bayes(NB), Support Vector Machine(SVM), and Extreme Gradient Boosting(XGboost) and Naïve Bayes(NB) that may predict diabetes based on early patient symptoms.

METHODOLOGY

The algorithmic procedure that is being suggested in this work and is shown in Figure 1. The first step is to use the data set as an input to the prediction algorithm, and the second step is to run the algorithm through the evaluation model, which is a way for creating a confusion matrix in order to assess the algorithm's accuracy in classifying data. After that, we arrive at the algorithm that accurately predicts diabetes.

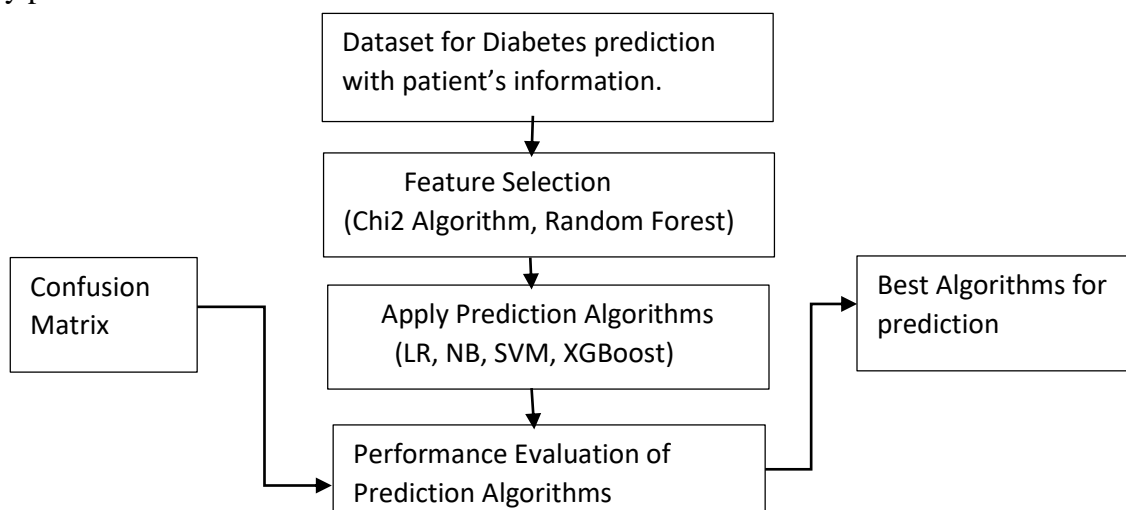


Fig 1. Model Architecture

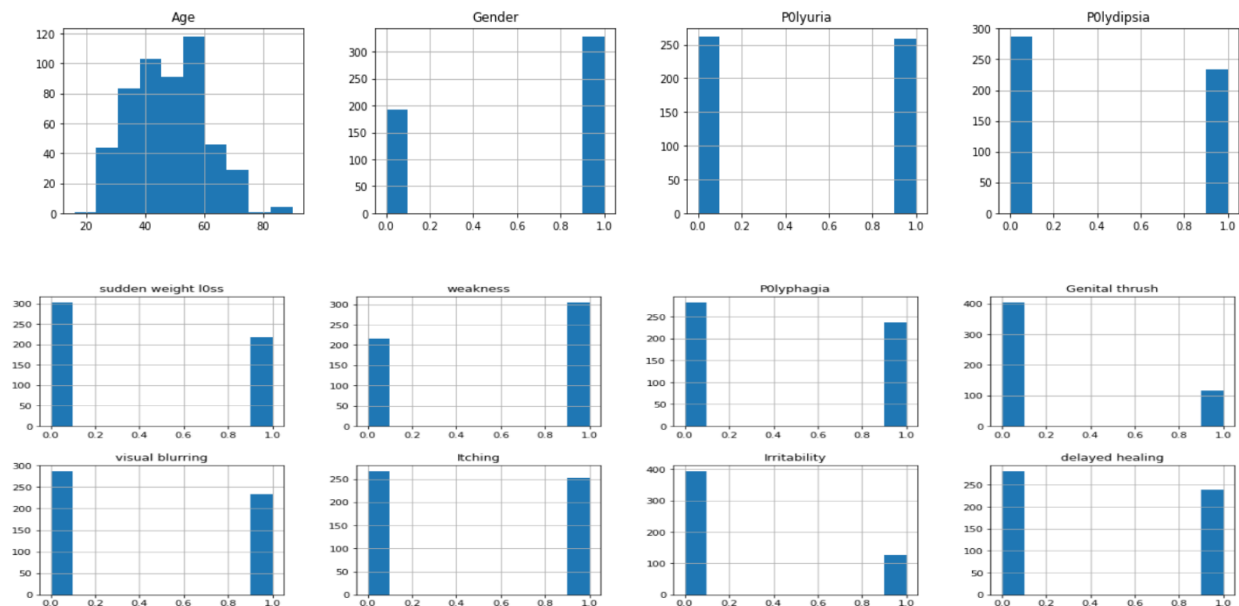
DATASET

The data set has been collected from the open source standard test data set website Kaggle Repository that consists the information of 520 patients. Following table shows the features/attributes of dataset.

Table 1 Description of attribute

	Attributes	Values
1	Age	16-90
2	Gender	1.Male, 0.Female
3	Polyuria	1.Yes, 0.No.
4	Polydipsia	1.Yes, 0.No.
5	Sudden weight loss	1.Yes, 0.No.
6	Weakness	1.Yes, 0.No.
7	Polyphagia	1.Yes, 0.No.
8	Genital thrush	1.Yes, 0.No.
9	Visual blurring	1.Yes, 0.No.
10	Itching	1.Yes, 0.No.
11	Irritability	1.Yes, 0.No.
12	Delayed healing	1.Yes, 0.No.
13	Partial paresis	1.Yes, 0.No.
14	Muscle stiffness	1.Yes, 0.No.
15	Alopecia	1.Yes, 0.No.
16	Obesity	1.Yes, 0.No.
17	Result	1.Positive, 0.Negative.

Histograms are graphs that display frequency distributions. The amount of observations in each interval is shown on this graph.



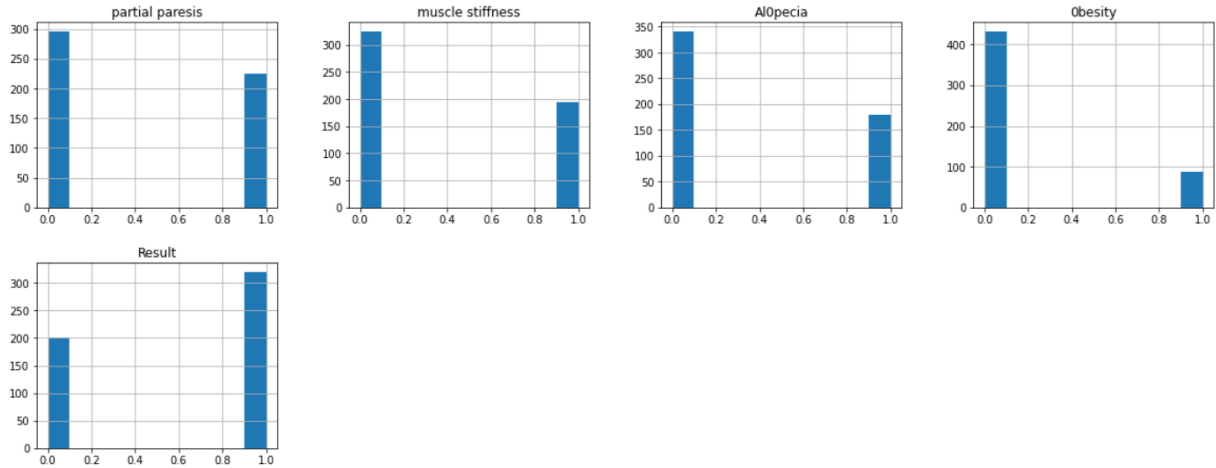


Fig 2. Data distribution using Histogram

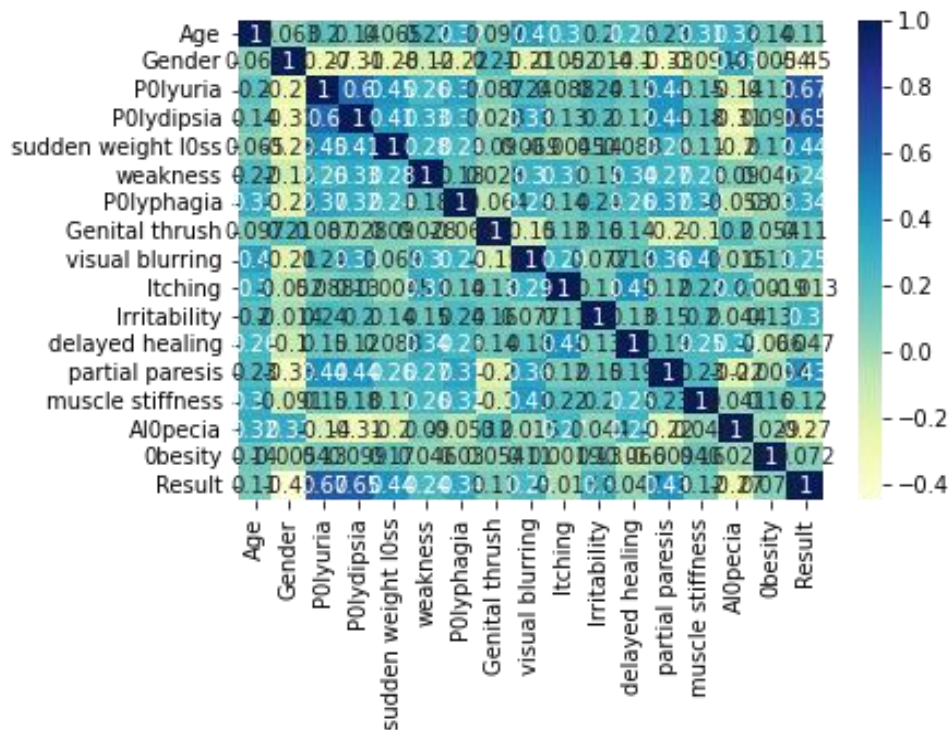
Correlation Matrix:

A correlation matrix is a table that contains the coefficients of correlation between variables. Each cell in the table displays the relationship between two variables. The value ranges from -1 to 1 Here,

-1 specifies a perfectly negative linear correlation between two variables.

0 indicates no linear correlation between two variables

1 indicates a perfectly positive linear correlation between two variables



Data Splitting: Data is split into two parts: a larger portion is used for training the model, and a smaller portion of data is used for testing purposes. Dataset has been split into 70:30 [Training data: Testing data]. Training data is used to train or fit the machine learning model. The test dataset is another subset of original data, which is independent of the training dataset.

```

▶ print(x_train.shape[0])
  print(y_test.shape[0])

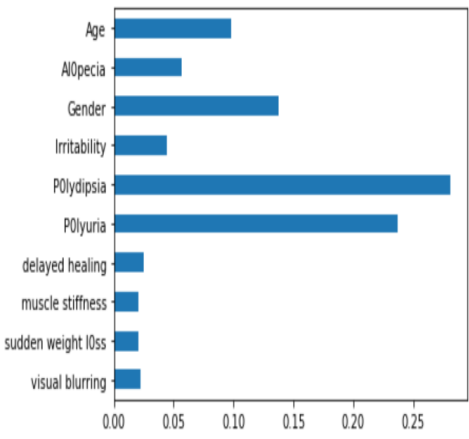
364
156

```

Feature Selection

When developing a predictive model, feature selection is the procedure of picking the characteristics from the dataset that contribute most to the predicted variable or output. Irrelevant features in the dataset may reduce the model's prediction efficiency as well as the classifiers' overall performance in terms of accuracy and complexity. The chi-square algorithm and the random forest algorithm have been used in this analysis.

Table 2. Selected Features

Feature selected by Chi2 Algorithm			Feature selected by Random Forest Algorithm	
	Features	Score		
3	Polydipsia	120.785515		
2	Polyuria	116.184593		
4	sudden weight loss	57.749309		
12	partial paresis	55.314286		
1	Gender	38.747637		
10	Irritability	35.334127		
6	Polyphagia	33.198418		
14	Alopecia	24.402793		
0	Age	18.845767		
8	visual blurring	18.124571		

The figure below depicts the correlation map with outcome after selecting features for which the ML algorithm will be used to predict diabetes.

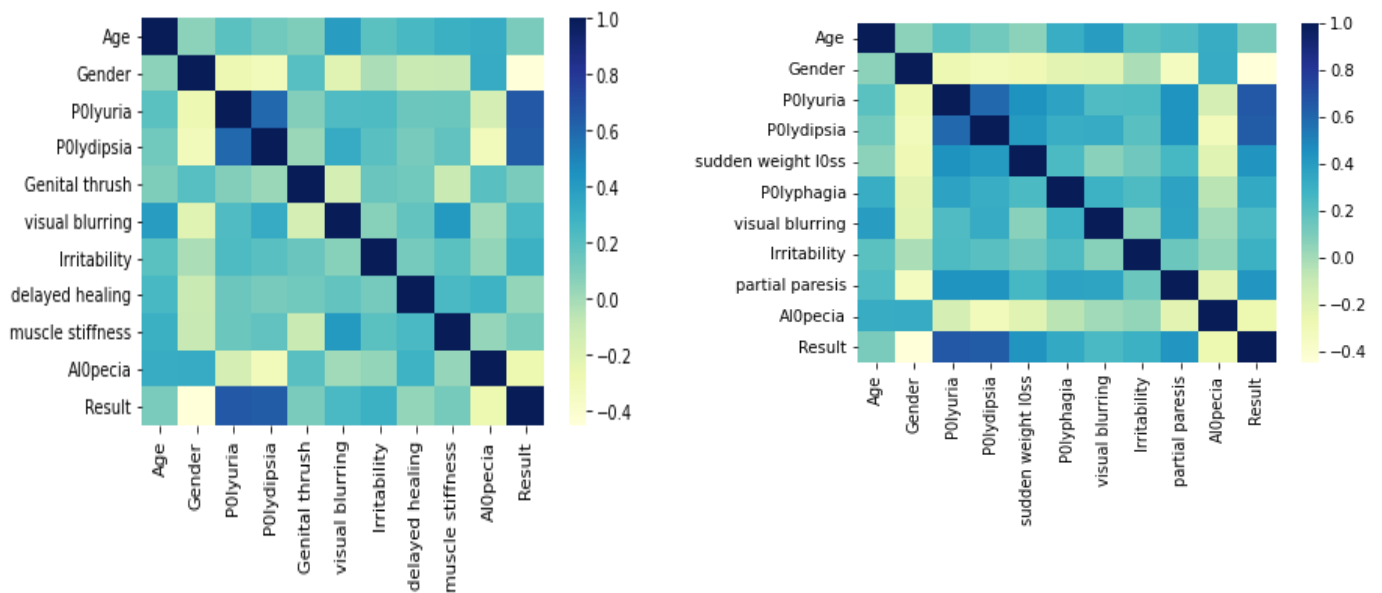


Fig 4. Correlation map using new features (Random Forest,Chi2 Algorithm)

Different evaluation metrics; confusion matrix, precision, accuracy and recall. are used to evaluate the performance and to find out the best algorithm.

Confusion Matrix: A Confusion matrix is a $N \times N$ matrix used to evaluate a classification model's performance, where N is the number of target classes. The matrix evaluates the actual expected values against the values estimated by the model using machine learning.

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Fig 5. Confusing matrix representation

- **(TP):** Outcome where the model correctly predicts the positive class.
- **(TN):** Outcome where the model correctly predicts the negative class.

- **(FP):** Also referred as a **type 1 error**, an outcome where the model incorrectly predicts the positive class when it is actually negative.
- **(FN):** Also referred as a **type 2 error**, an outcome where the model incorrectly predicts the negative class when it is actually positive

Accuracy

This is simply equal to the proportion of predictions that the model classified correctly.

$$Accuracy = \frac{\# \text{ of correct predictions}}{\text{total \# of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Precision tells us how many of the correctly predicted cases actually turned out to be positive.

Here's how to calculate Precision:

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

This would determine whether our model is reliable or not.

Precision: Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

And here's how we can calculate Recall:

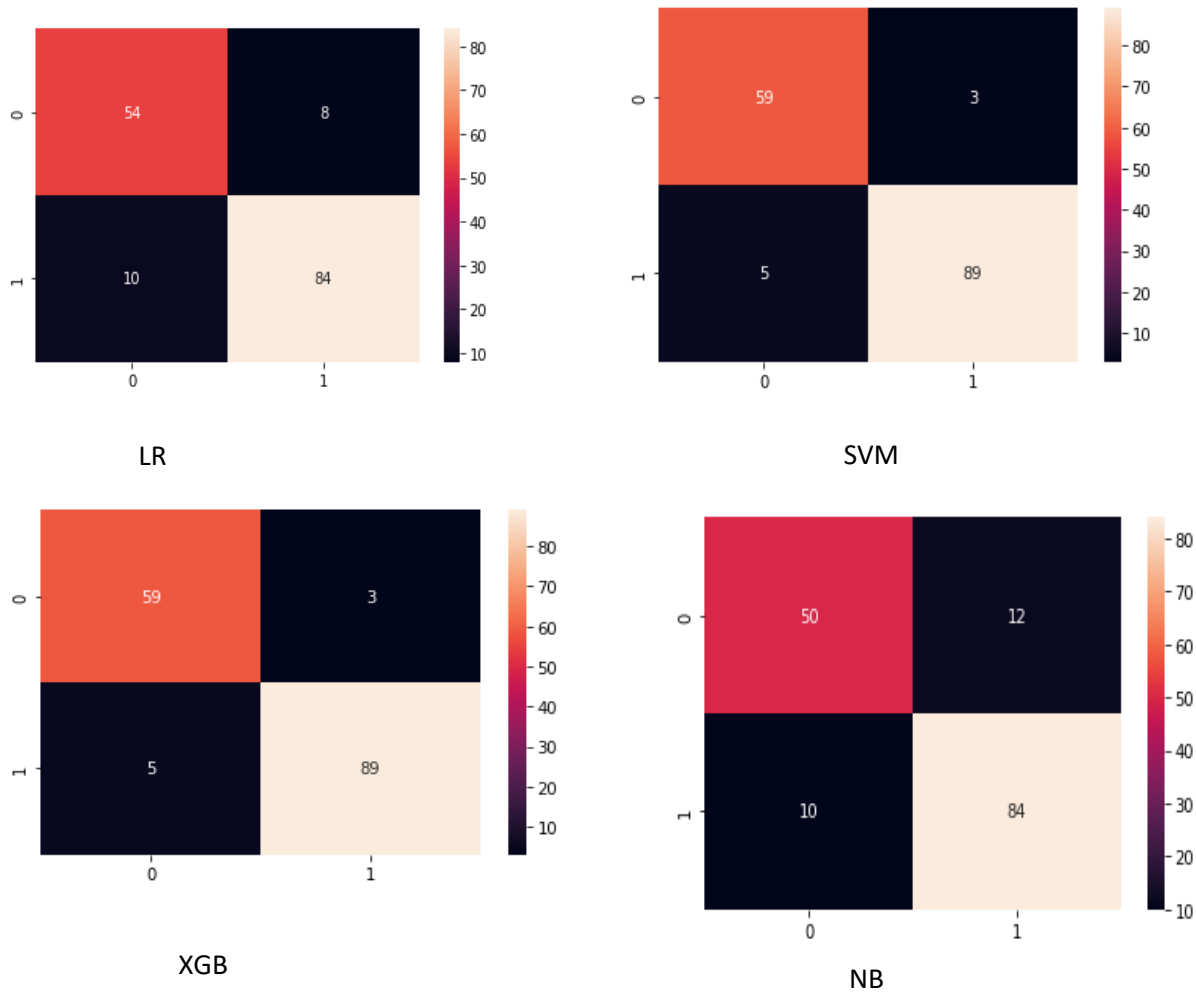
$$Recall = \frac{TP}{TP + FN}$$

Result Analysis:

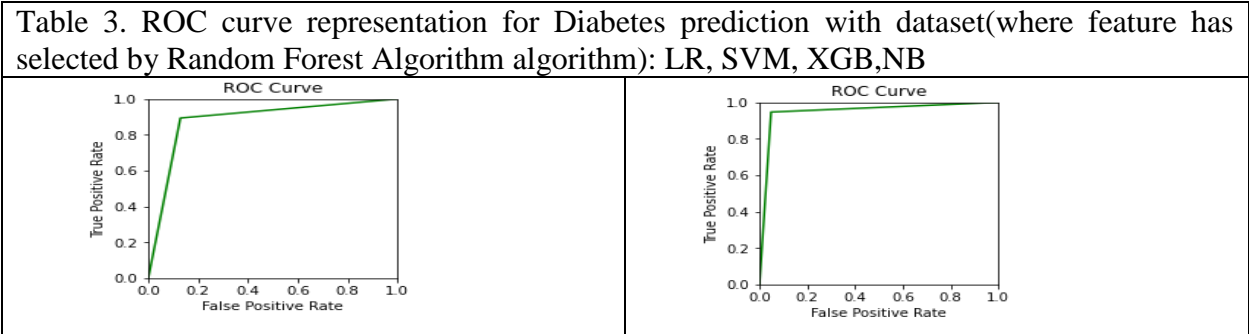
Table 4. Analysis with Feature selection using Random Forest algorithm.				
ML Algorithms	Accuracy	Precision	Recall	True positive(TP) TN(True Negative) FP(false Positive) FN(False Negative)
LR	89%	91%	89%	TP = 54 TN= 84 FP = 8 FN = 10
SVM	95%	97%	95%	TP = 59 TN= 89 FP = 3 FN = 5
XGB	95%	97%	95%	TP = 59 TN= 89 FP = 3 FN = 5

NB	86%	88%	89%	TP = 50 TN= 84 FP = 12 FN = 10
----	-----	-----	-----	---

Confusion Matrix for 4 algorithms.



Roc Curve: The ROC curve can be employed to establish a threshold for a classifier that optimizes true positives while minimizing false positives. ROC Curves aid in determining the precise trade-off between the true positive rate and the false positive rate for a model utilizing various probability threshold metrics.



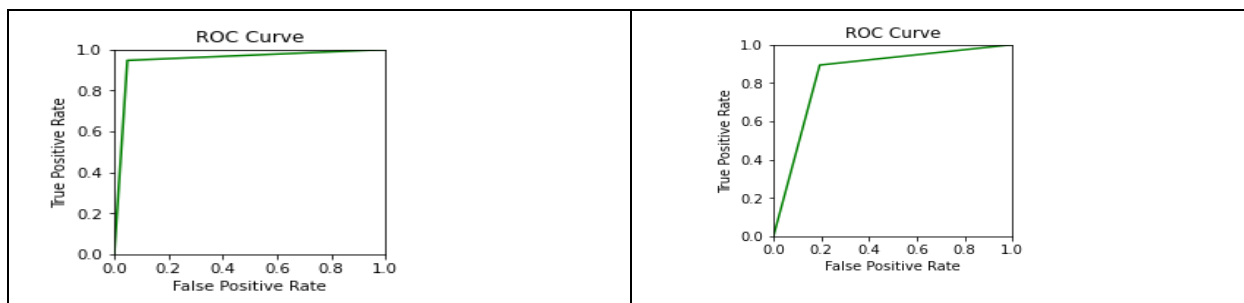


Table 5. Analysis with Feature selection using chi2 square algorithm.				
ML Algorithms	Accuracy	Precision	Recall	True positive(TP) TN(True Negative) FP(false Positive) FN(False Negative)
LR	91%	92%	94%	TP = 54 TN= 88 FP = 8 FN = 6
SVM	93%	94%	95%	TP = 57 TN= 89 FP = 5 FN = 5
XGB	94%	95%	96%	TP = 57 TN= 90 FP = 5 FN = 4
NB	88%	0.87879	92%	TP = 50 TN= 87 FP = 12 FN = 7

Confusion Matrix for 4 algorithms

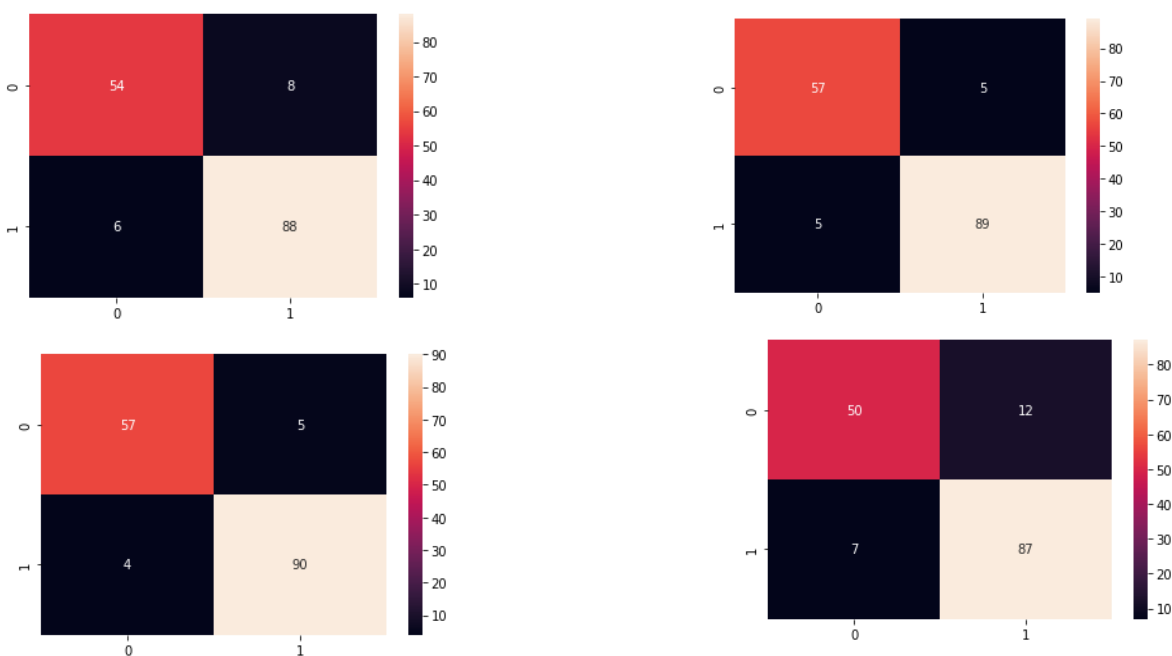
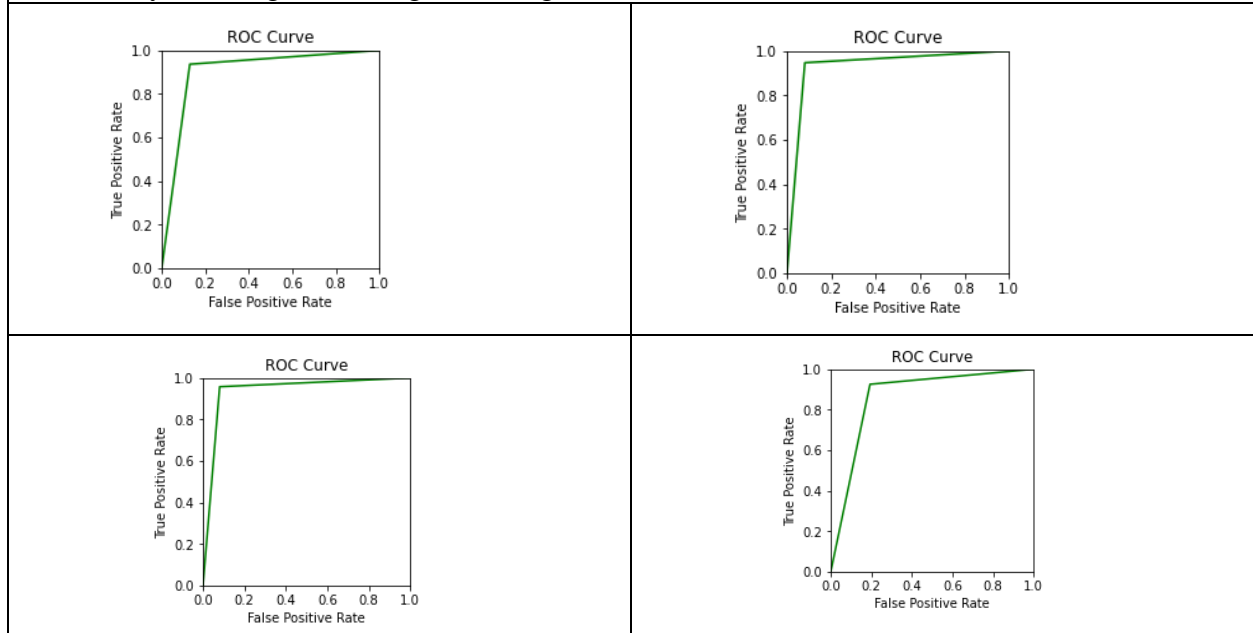


Fig 6. Confusing matrix representation of (Logistic Regression, Support Vector Machine, Extreme Gradient Boosting(XGboost) and Naïve Bayes(NB))

Table 6. ROC curve representation for Diabetes prediction with dataset(where feature has selected by Chi2 algorithm Algorithm algorithm): LR, SVM, XGB,NB



Conclusion

Here, the analysis shows that, Extreme Gradient Boosting(XGboost) algorithm shows highest accuracy through the confusion matrix evaluation test as well as in ROC curve when dataset used with feature selected by Chi2 algorithm. And, both Support Vector Machine (SVM) & Extreme Gradient Boosting(XGboost) algorithm shows highest accuracy through the confusion matrix evaluation test as well as in ROC curve when dataset used with feature selected by Random Forest algorithm.

References:

1. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
2. https://scikit-learn.org/stable/modules/naive_bayes.html
3. <https://scikit-learn.org/stable/modules/svm.html>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>
5. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.chi2.html
6. <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>

Code:

```
File Edit View Insert Runtime Tools Help Last edited on December 6
```

```
+ Code + Text
```

```
[ ] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive
```

```
[ ] data = pd.read_csv('/content/drive/My Drive/Dataset/diabetes_dataset.csv')
data
```

	Age	Gender	P0lyuria	P0lydipsia	sudden weight loss	weakness	P0lyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Al0pecia	Obesity	Result
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	1
1	58	1	0	0	0	1	0	0	1	0	0	0	1	1	0	1	1
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	1
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	1
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
...
516	39	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	1
516	48	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1
517	58	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1	1
518	32	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0	0
519	42	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

520 rows x 17 columns

DATA SET SELECTION

```
File Edit View Insert Runtime Tools Help Last edited on December 6
```

```
+ Code + Text
```

```
[ ] import pandas as pd
import numpy as np
x = data.iloc[:,0:15] #independent columns
y = data.iloc[:,16]
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=10)
rf.fit(x_train, y_train)
sort = rf.feature_importances_
sort
p = pd.Series(rf.feature_importances_, index=x.columns)
p.nlargest(10).sort_index(ascending=False).plot(kind='barh')

#p.nlargest(10).plot(kind='barh')
plt.show()
```

```
File Edit View Insert Runtime Tools Help Last edited on December 6
```

```
+ Code + Text
```

```
[ ] from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
#print(dfscores)
#dfcolumns =pd.DataFrame(x.columns.values.tolist())
dfcolumns =pd.DataFrame(data.columns.values.tolist())

#b=data.columns.values.tolist()
#print(b)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Features','Score'] #naming the dataframe columns
print(featureScores.nlargest(10,'Score')) #print 10 best features
```

	Features	Score
3	P0lydipsia	120.785515
2	P0lyuria	116.184593
4	sudden weight loss	57.749309
12	partial paresis	55.314286
1	Gender	38.747637
10	Irritability	35.334127
6	P0lyphagia	33.198418
14	Al0pecia	24.402793
0	Age	18.845767
8	visual blurring	18.124571

```
[ ] import pandas as pd
import numpy as np
x = data.iloc[:,0:15] #independent columns
```

Feature Selection codes

Code after Feature selection using Chi2 Algorithm for Diabetes Prediction using LR, SVM, XGB and NB

```
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix

classifier= LogisticRegression()
classifier.fit(x_train, y_train)
predictions = classifier.predict(x_test)

cf_matrix=confusion_matrix(y_test,predictions)
print(cf_matrix)

print('\nTrue Positives(TP) = ', cf_matrix[0,0])
print('True Negatives(TN) = ', cf_matrix[1,1])
print('False Positives(FP) = ', cf_matrix[0,1])
print('False Negatives(FN) = ', cf_matrix[1,0])
print('\n')
#print(classification_report(y_test,predictions))

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

print('Precision: %.5f' % metrics.precision_score(y_test, predictions))
print('Recall: %.5f' % metrics.recall_score(y_test, predictions))

print('Accuracy: %.3f' % metrics.accuracy_score(y_test, predictions))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, predictions)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, predictions)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
```

```
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(x_train, y_train)
svc_pred = svc_model.predict(x_test)

from sklearn import metrics
print("Accuracy Score =", format(metrics.accuracy_score(y_test, svc_pred)))
from sklearn.metrics import classification_report, confusion_matrix
cf_matrix=confusion_matrix(y_test, svc_pred)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

print('Precision: %.5f' % metrics.precision_score(y_test,svc_pred))
print('Recall: %.5f' % metrics.recall_score(y_test,svc_pred))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, svc_pred)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, svc_pred)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```

colab.research.google.com/drive/12jihBs8nmp-LRbcNVt55Asz2BSN4_i3A#scrollTo=-rP2Qx_1slf3

diabtecs_dataset_Ch2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 12:17 AM

+ Code + Text

```
[40] from xgboost import XGBClassifier

xgb_model = XGBClassifier(gamma=0)
xgb_model.fit(x_train, y_train)

from sklearn import metrics
xgb_pred = xgb_model.predict(x_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test, xgb_pred)))

from sklearn.metrics import classification_report, confusion_matrix
cf_matrix=confusion_matrix(y_test, xgb_pred)
print(cf_matrix)
import seaborn as sns
sns.heatmap(cf_matrix, annot=True)
print('Precision: %.5f' % metrics.precision_score(y_test, xgb_pred))
print('Recall: %.5f' % metrics.recall_score(y_test, xgb_pred))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, xgb_pred)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, xgb_pred)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```

Accuracy Score = 0.8432876073876073

Type here to search

12:53 AM 08-Dec-22

colab.research.google.com/drive/12jihBs8nmp-LRbcNVt55Asz2BSN4_i3A#scrollTo=p7NyrOl8sWNu

diabtecs_dataset_Ch2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 12:17 AM

+ Code + Text

```
[41] from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
import seaborn as sns
sns.heatmap(cm, annot=True)
ac = accuracy_score(y_test, y_pred)
print(ac)

print('Precision: %.5f' % metrics.precision_score(y_test, y_pred ))
print('Recall: %.5f' % metrics.recall_score(y_test, y_pred ))

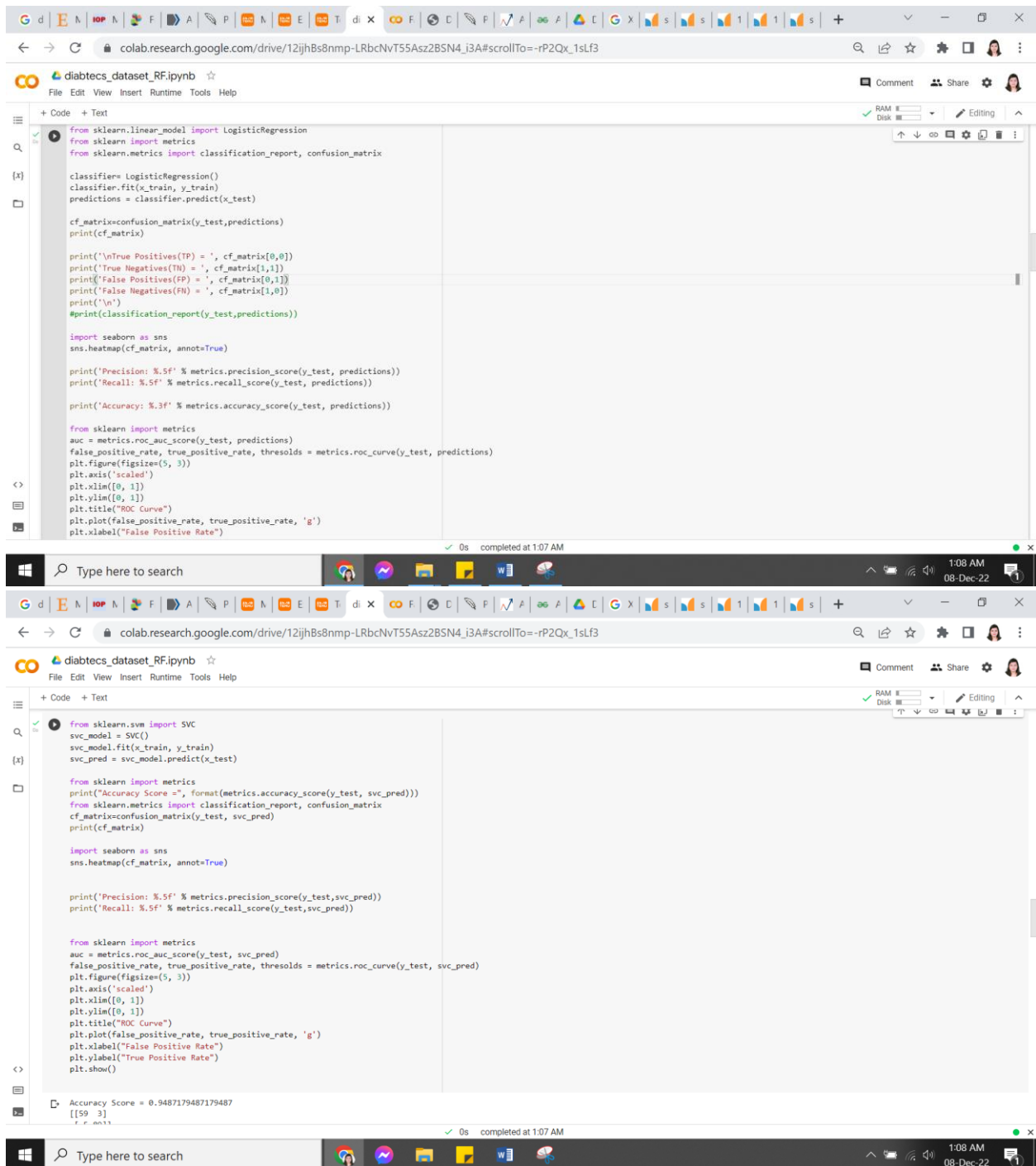
print('Accuracy: %.3f' % metrics.accuracy_score(y_test, y_pred ))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, y_pred)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, y_pred)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```

Type here to search

12:53 AM 08-Dec-22

Code after Feature selection using Random Forest Algorithm for Diabetes Prediction using LR, SVM, XGB and NB



The image displays two screenshots of a Google Colab notebook titled "diabtcets_dataset_RF.ipynb". The notebook is open in a web browser, showing the code editor and the output area.

Top Screenshot: The code defines a Logistic Regression classifier and evaluates its performance. It includes imports for LogisticRegression, metrics, classification_report, and confusion_matrix. The classifier is trained on x_train and y_train, and predictions are made on x_test. A confusion matrix is printed, followed by True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). A classification report is also printed. The code then imports seaborn and creates a heatmap of the confusion matrix. Finally, it prints Precision, Recall, and Accuracy scores, and generates an ROC curve plot.

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix

classifier= LogisticRegression()
classifier.fit(x_train, y_train)
predictions = classifier.predict(x_test)

cf_matrix=confusion_matrix(y_test,predictions)
print(cf_matrix)

print('\nTrue Positives(TP) = ', cf_matrix[0,0])
print('\nTrue Negatives(TN) = ', cf_matrix[1,1])
print('\nFalse Positives(FP) = ', cf_matrix[0,1])
print('\nFalse Negatives(FN) = ', cf_matrix[1,0])
print('\n')
#print(classification_report(y_test,predictions))

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

print('Precision: %.5f' % metrics.precision_score(y_test, predictions))
print('Recall: %.5f' % metrics.recall_score(y_test, predictions))

print('Accuracy: %.3f' % metrics.accuracy_score(y_test, predictions))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, predictions)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, predictions)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

Bottom Screenshot: The code defines an SVM classifier and evaluates its performance. It includes imports for SVC, metrics, classification_report, and confusion_matrix. The classifier is trained on x_train and y_train, and predictions are made on x_test. A confusion matrix is printed, followed by True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). A classification report is also printed. The code then imports seaborn and creates a heatmap of the confusion matrix. Finally, it prints Precision, Recall, and Accuracy scores, and generates an ROC curve plot.

```
from sklearn.svm import SVC
svc_model = SVC()
svc_model.fit(x_train, y_train)
svc_pred = svc_model.predict(x_test)

from sklearn import metrics
print("Accuracy Score = ", format(metrics.accuracy_score(y_test, svc_pred)))
from sklearn.metrics import classification_report, confusion_matrix
cf_matrix=confusion_matrix(y_test, svc_pred)
print(cf_matrix)

import seaborn as sns
sns.heatmap(cf_matrix, annot=True)

print('Precision: %.5f' % metrics.precision_score(y_test, svc_pred))
print('Recall: %.5f' % metrics.recall_score(y_test, svc_pred))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, svc_pred)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, svc_pred)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()
```

The output area shows the Accuracy Score: 0.9487179487179487.

```
colab.research.google.com/drive/12ijhBs8nmp-LRbcNVT55Asz2BSN4_i3A#scrollTo=-rP2Qx_1slf3

diabtecs_dataset_RF.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text
RAM 100% Disk 100% Editing

from xgboost import XGBClassifier

xgb_model = XGBClassifier(gamma=0)
xgb_model.fit(x_train, y_train)

from sklearn import metrics
xgb_pred = xgb_model.predict(x_test)
print("Accuracy Score =", format(metrics.accuracy_score(y_test, xgb_pred)))

from sklearn.metrics import classification_report, confusion_matrix
cf_matrix=confusion_matrix(y_test, xgb_pred)
print(cf_matrix)
import seaborn as sns
sns.heatmap(cf_matrix, annot=True)
print('Precision: %.5f' % metrics.precision_score(y_test, xgb_pred))
print('Recall: %.5f' % metrics.recall_score(y_test, xgb_pred))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, xgb_pred)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, xgb_pred)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()

Accuracy Score = 0.9487179487179487
[[59  3]
```

```
colab.research.google.com/drive/12ijhBs8nmp-LRbcNVT55Asz2BSN4_i3A#scrollTo=-rP2Qx_1slf3

diabtecs_dataset_RF.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text
RAM 100% Disk 100% Editing

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
import seaborn as sns
sns.heatmap(cm, annot=True)
ac = accuracy_score(y_test, y_pred)
print(ac)

print('Precision: %.5f' % metrics.precision_score(y_test, y_pred))
print('Recall: %.5f' % metrics.recall_score(y_test, y_pred))

print('Accuracy: %.3f' % metrics.accuracy_score(y_test, y_pred))

from sklearn import metrics
auc = metrics.roc_auc_score(y_test, y_pred)
false_positive_rate, true_positive_rate, thresholds = metrics.roc_curve(y_test, y_pred)
plt.figure(figsize=(5, 3))
plt.axis('scaled')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.title("ROC Curve")
plt.plot(false_positive_rate, true_positive_rate, 'g')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.show()

[[50 12]
 [10 84]]
0.8589743589743589
Precision: 0.87500
```