BIGDATAANALYTICS

Exploring the Effectiveness of Dimensionality Reduction using Bigdata

Team Members:
Swetha Kanakavalli (811198105)
Manohar Kanderaboina (811225263)
Alekhya kari (811255515)

INTRODUCTION:

Big data is becoming increasingly important in clinical and medical research because it allows for the examination and interpretation of massive volumes of patient data. Healthcare institutions and research institutes today have greater access to data than ever before, including genetic and electronic medical records. Many medical disorders, including rare diseases and complex conditions, can be identified, and treated using this data. Healthcare workers can find patterns and trends in patient data that would be impossible to find using conventional analytical procedures by utilizing big data analytics techniques.As a result, big data is rapidly transforming the way medical research is conducted and providing new views on the diagnosis and treatment of a wide range of medical disorders.

PROBLEM DESCRIPTION:

This project elaborates the use of big data and learning in the health industry, specifically in the diagnosis of different medical datasets, with a focus on pregnancy-related testing for women using Cardiotocography (CTG). The CTG scan measures the fetal heart rate and the levels of signals from the mother's uterine contractions to identify the fetus as "Normal," "Suspect," or "Pathologic." The data, on the other hand, has a high number of attributes/dimensions that may be reduced using dimensionality reduction techniques like as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) to train the classifier model accurately and quickly.

In the field of medical diagnostics, such as clinical trials, where a large number of variables are regularly collected from patients, dimensionality reduction techniques like as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) have various real-world applications. PCA and LDA may be used to identify relevant variables and reduce the dimensionality of a dataset, which can aid in identifying the most essential elements determining therapy efficacy. Furthermore, in Electronic Medical Records (EMRs), patient data is frequently gathered throughout time and across numerous factors, resulting in high-dimensional datasets.

PCA and LDA may be used to decrease the dimensionality of these datasets and find relevant factors linked with a certain disease or condition.This can serve to enhance the accuracy of diagnosis and prognosis, as well as support clinical decision making.

BACKGROUND:

Because of the huge number of variables or characteristics that might be included in such datasets, dimensionality reduction techniques have become more relevant in big data research. Principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), and non-negative matrix factorization are some common dimensionality reduction techniques (NMF).

Several studies on the analysis of dimensionality reduction techniques on big data have been conducted. Khairnar et al"Comparative.'s Analysis of Dimensionality Reduction Techniques for High-Dimensional Data" is one such study (2020). The authors of this study assessed the performance of six different dimensionality reduction techniques on high-dimensional data. Among the approaches investigated were PCA, t-SNE, NMF, linear discriminant analysis (LDA), autoencoder-based methods, and random projection. The authors came to the conclusion that no single strategy consistently outperformed the others and that the technique used is determined by the unique characteristics of the dataset and the job at hand. Another study is "Dimensionality Reduction Techniques for Big Data: A Review" by Vatsa et al. (2019). The authors of this article offered an overview of several dimensionality reduction approaches and their relevance to big data. The authors emphasized the difficulties involved with applying these approaches to huge data, such as computational complexity, scalability, and interpretability. The paper also highlighted current advances in dimensionality reduction strategies for big data, such as deep learning-based approaches.

Aside from these research, a plethora of additional publications have investigated dimensionality reduction strategies for huge data. Typically, these studies compare and assess the performance of various approaches on distinct datasets. Overall, big data dimensionality reduction approach analysis is an active area of study, and continuous improvements are anticipated to increase the capacity to analyze and extract insights from vast and complicated datasets.

PROBLEM DEFINITION:

The primary goals of this project are to review and analyze the effectiveness of PCA and LDA on machine learning algorithms using various metrics, to ensure that dimensionality reduction approaches do not impair ML algorithm performance, to use feature engineering methods to optimize ML performance bottlenecks, and to determine the approach's effectiveness on datasets with varying dimensionalities.

The topic is about the challenges of applying dimensionality reduction techniques to vast volumes of data, such as the computational cost of the algorithms, the difficulty in picking the suitable hyperparameters, and the interpretability of the findings. In order to design effective and scalable dimensionality reduction methods for large data analysis, a variety of challenges must be overcome.

PROPOSED TECHNIQUES:

In machine learning, two extensively used dimensionality reduction approaches are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). PCA is an unsupervised approach for reducing dataset dimensionality by finding the most essential variables or features that contribute to the total variance in the data. LDA, on the other hand, is a supervised strategy that seeks the most discriminative characteristics that distinguish between the various classes in a dataset.

The CTG dataset, also known as cardiotocography data, as well as PCA and LDA, may be used to determine the most critical factors affecting the health of the fetus throughout pregnancy. The dataset comprises data on numerous fetal heart rate patterns as well as other clinical characteristics that might be utilized to predict the likelihood of fetal distress.

We implemented coding PCA and LDA using different python libraries which typically involved loading the dataset, preprocessing the data, and applying the PCA or LDA algorithm to reduce the dimensionality of the data.

Following completion of the coding section, a detailed study of the data is required to comprehend the performance of the dimensionality reduction strategies. This entails evaluating the abilities of PCA and LDA to decrease the dimensionality of data while maintaining the most critical information. Additionally, a review of the relevant literature on dimensionality reduction approaches for the CTG dataset can provide insights into best practices and potential improvements to the analysis.

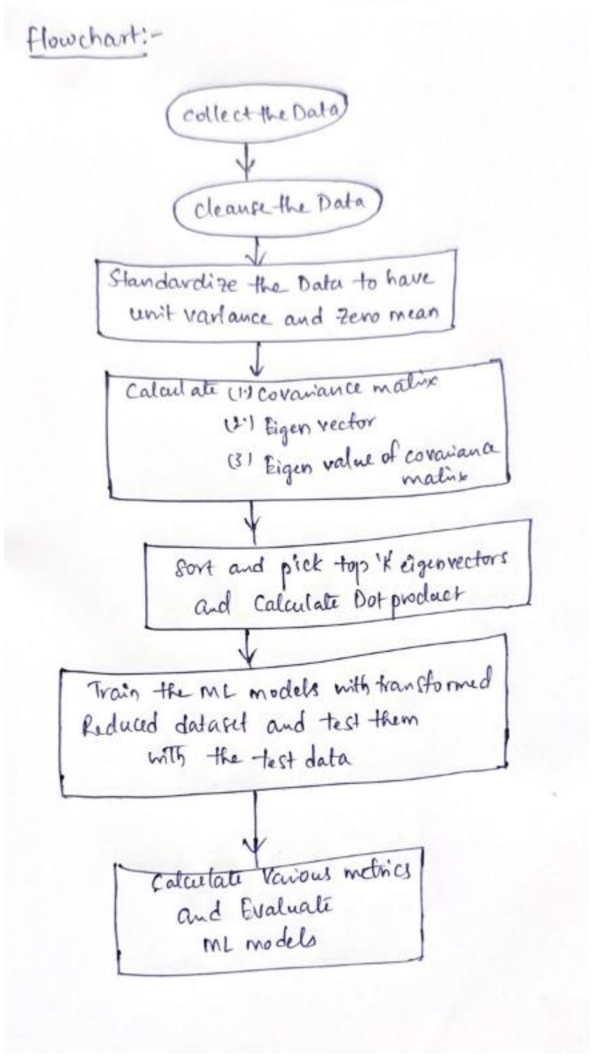
Features of CTG Dataset:

The Cardiotocography (CTG) dataset includes fetal heart rate (FHR) patterns and other clinical indicators that may be utilized to determine a fetus's health throughout pregnancy. The following are some of the CTG dataset's key features:

1. Baseline FHR: It is a significant measure of fetal well-being and refers to the average FHR during a 10-minute period.
2. FHR Variability: It monitors FHR variations and is a major indicator of fetal distress.
3. Accelerations: They are brief increases in the FHR that indicate fetal well-being. Decelerations: They are brief decreases in the FHR that can indicate fetal distress. Presence of uterine contractions: They can affect the FHR and can be an important factor in assessing fetal health.
4. Gestational age: It is the age of the fetus in weeks and can provide insights into the development of the fetus.
5. pH of fetal scalp blood: It is an invasive test that measures the acidity of the fetal scalp blood and can provide information about the fetal well-being.

6. Presence of meconium: It is the first stool of the newborn and its presence in the amniotic fluid can indicate fetal distress.

VISUAL APPLICATIONS:


Flowchart:-
collect the Data
Cleanse the Data
Standardize the Data to have unit variance and Zero mean
Calculate (1) covariance matrix (2) Eigen vector (3) Eigen value of covariance matrix
Sort and pick top 'K' eigenvectors and Calculate Dot product
Train the ML models with transformed Reduced dataset and test them with the test data
Calculate Various metrics and Evaluate ML models

EXPERIMENTAL EVALUATION:

Dataset:

Cardiotocography (CTG) is a medical procedure that monitors the fetal heart rate and uterine contractions throughout pregnancy and delivery. The fetal heart rate is monitored using an electronic device called a fetal heart rate monitor, which captures the rate and rhythm of the fetal heartbeats. A tocodynamometer is used to record the frequency, length, and strength of uterine contractions.

The cardiotocography dataset has 2126 fetal cardiotocograms that were automatically processed, and the respective diagnostic features measured. The CTGs were also classified by three expert obstetricians and a consensus classification label assigned to each of them.

Detail Design of Features:

Raw Data format:

Column Name	Description
Filename	of CTG examination
Date	of the examination
b	start instant
e	end instant
LBE	baseline value (medical expert)
LB	baseline value (SisPorto)
AC	accelerations (SisPorto)
FM	fetal movement (SisPorto)
UC	uterine contractions (SisPorto)
ASTV	percentage of time with abnormal short-term variability (SisPorto)
mSTV	mean value of short-term variability (SisPorto)
ALTV	percentage of time with abnormal long-term variability (SisPorto)
mLTV	mean value of long-term variability (SisPorto)
DL	light decelerations
DS	severe decelerations
DP	prolonged decelerations
DR	repetitive decelerations
Width	histogram width
Min	low freq. of the histogram
Max	high freq. of the histogram
Nmax	number of histogram peaks
Nzeros	number of histogram zeros
Mode	histogram mode
Mean	histogram mean
Median	histogram median
Variance	histogram variance

Tendency	histogram tendency: -1=left assymetric; 0=symmetric; 1=right assymetric
A	calm sleep
B	REM sleep
C	calm vigilance
D	active vigilance
SH	shift pattern (A or Susp with shifts)
AD	accelerative/decelerative pattern (stress situation)
DE	decelerative pattern (vagal stimulation)
LD	largely decelerative pattern
FS	flat-sinusoidal pattern (pathological state)
SUSP	suspect pattern
CLASS	Class code (1 to 10) for classes A to SUSP
NSP	Normal=1; Suspect=2; Pathologic=3

Features/ Dimensions considered for Dimensionality reduction.

Features
LB
AC
FM
UC
DL
DS
DP
ASTV
MSTV
ALTV
MLTV
Width
Min
Max
Nmax
Nzeros
Mode
Mean
Median
Variance
Tendency

Target Label:

NSP

ANALYSIS:

CTG dataset gives an abundance of information for predicting diseases in infants before birth or in the womb. To provide accurate predictions, this dataset must first be processed before being fed into an ML classifier model. Because the dataset contains a diverse set of attributes/dimensions, it must be preprocessed, which is where dimensionality reduction techniques come in to map high dimensionality correlated attributes to lower dimensionality unit variance low correlated dimensions.

Feature engineering is also important in extracting the converted dimension/feature from raw data and converting it into actual input to the ML model.
Principal Component Analysis (PCA) and Linear Dimension Analysis (LDA) are two popular dimensionality reduction techniques that are tested on some well-known ML algorithms such as Decision Tree, Nave Bayes, Random Forest, and Support Vector Machine (SVM) with the CTG dataset from the UCI Machine Learning repository. The experiment is then performed on datasets of varying dimensions, such as Diabetic Retinopathy and Intrusion Detection. The performance of the ML algorithms on these distinct datasets is measured before and after dimensionality reduction, with special emphasis focused on the performance of PCA vs LDA against several standard metrics such as F1-score, Accuracy, Sensitivity, Precision, Recall, and Specificity.
The goal of this investigation is to uncover and demonstrate how dimensionality reduction approaches increase the speed and performance of machine learning algorithms.

Implementation:

The following steps are followed to implement Dimensionality reduction and feed the transformed data into the ML model.

- We started Cleansing the dataset.
- Later we Standardized the dataset to have unit variance and zero mean and then
- We Calculated the co-variance matrix, the eigen vector and eigen value of the co-variance matrix.
- Later we Sorted and picked up the top k eigen vectors and found the dot product with the raw data, essentially mapping the high dimensional raw data on to the k-dimensional transformed data producing the dimensionally reduced dataset.
- Trained the ML models with the new transformed reduced dataset and tested them with test data.

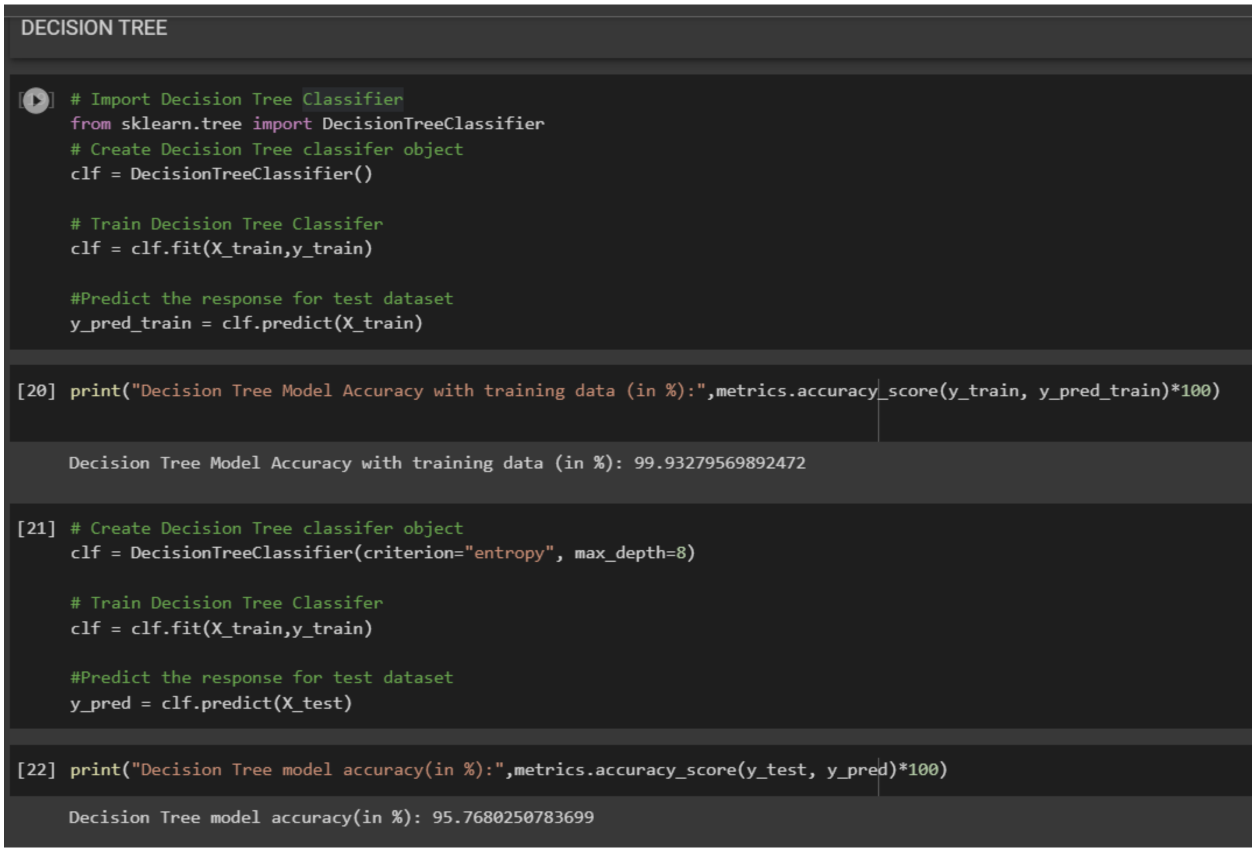
- Then we calculated the various metrics and evaluated the ML models

- We Performed step 6 & 7 but now without dimensionality reduction and then we compared.

Preliminary results:

We have completed performing LDA, PCA for the CTG dataset and run the 4 ML classifiers on both the reduced datasets. For comparison purposes we have also run the ML models on full non-reduced dataset. Preliminary results are attached below.

PCA:



DECISION TREE

```
# Import Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred_train = clf.predict(X_train)
```

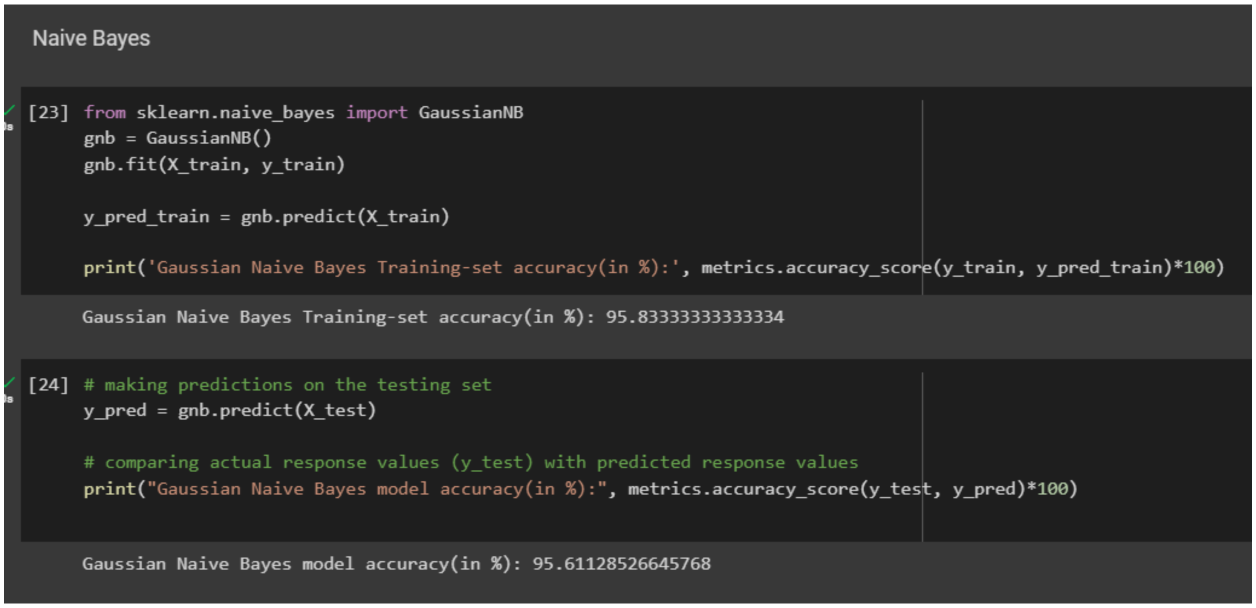
```
[20] print("Decision Tree Model Accuracy with training data (in %):",metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Decision Tree Model Accuracy with training data (in %): 99.93279569892472
```

```
[21] # Create Decision Tree classifer object
     clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

     # Train Decision Tree Classifer
     clf = clf.fit(X_train,y_train)

     #Predict the response for test dataset
     y_pred = clf.predict(X_test)
```

```
[22] print("Decision Tree model accuracy(in %):",metrics.accuracy_score(y_test, y_pred)*100)
```

```
Decision Tree model accuracy(in %): 95.7680250783699
```



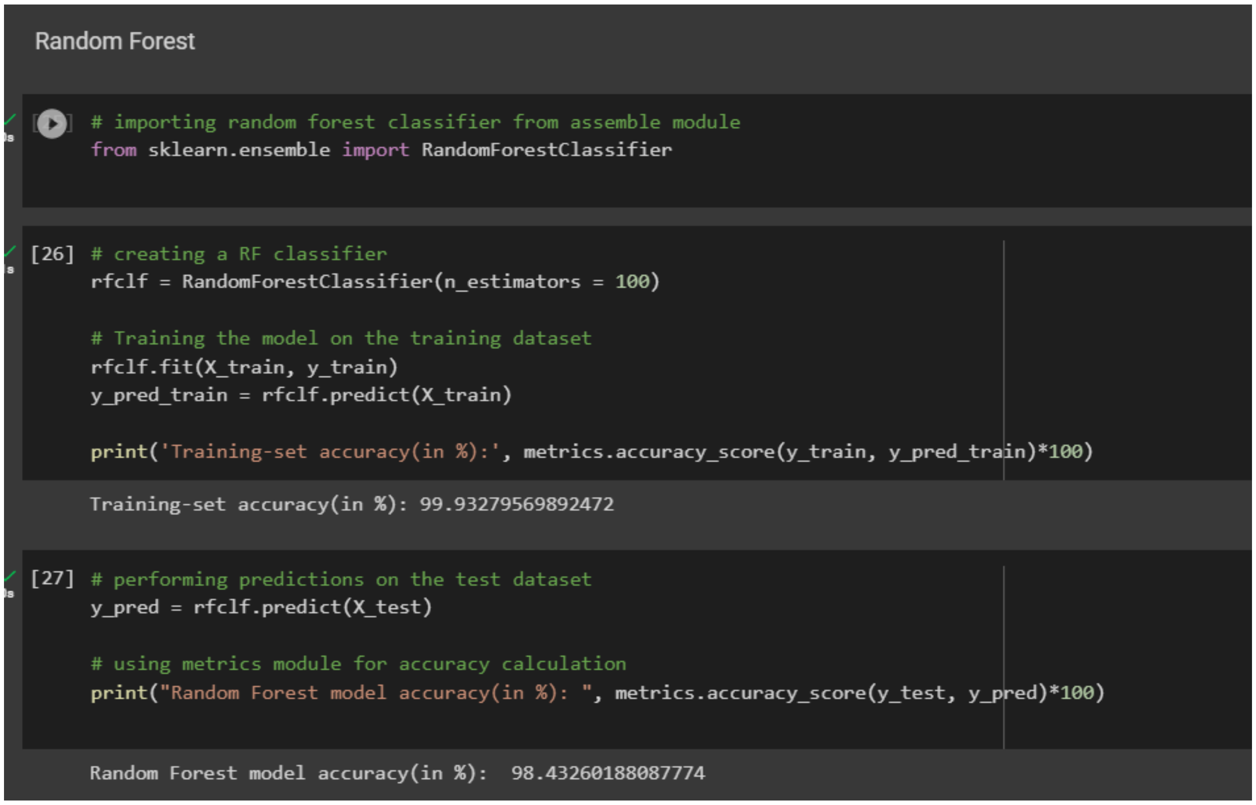
Naive Bayes

```
[23] from sklearn.naive_bayes import GaussianNB
     gnb = GaussianNB()
     gnb.fit(X_train, y_train)

     y_pred_train = gnb.predict(X_train)

     print('Gaussian Naive Bayes Training-set accuracy(in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Gaussian Naive Bayes Training-set accuracy(in %): 95.83333333333334
```

```
[24] # making predictions on the testing set
     y_pred = gnb.predict(X_test)

     # comparing actual response values (y_test) with predicted response values
     print("Gaussian Naive Bayes model accuracy(in %):", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Gaussian Naive Bayes model accuracy(in %): 95.61128526645768
```



Random Forest

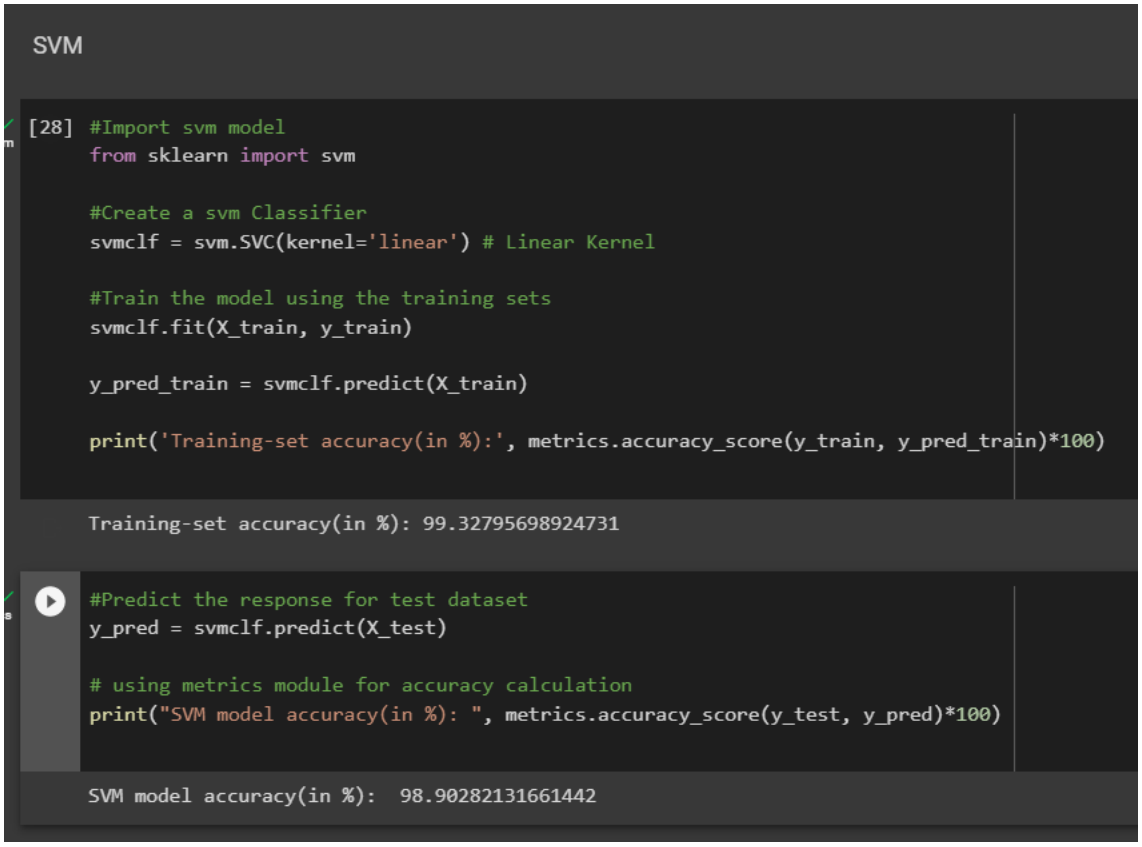
```
# importing random forest classifier from assemble module
from sklearn.ensemble import RandomForestClassifier
```

```
[26] # creating a RF classifier
     rfclf = RandomForestClassifier(n_estimators = 100)

     # Training the model on the training dataset
     rfclf.fit(X_train, y_train)
     y_pred_train = rfclf.predict(X_train)

     print('Training-set accuracy(in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Training-set accuracy(in %): 99.93279569892472
```

```
[27] # performing predictions on the test dataset
     y_pred = rfclf.predict(X_test)

     # using metrics module for accuracy calculation
     print("Random Forest model accuracy(in %): ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
Random Forest model accuracy(in %):  98.43260188087774
```



SVM

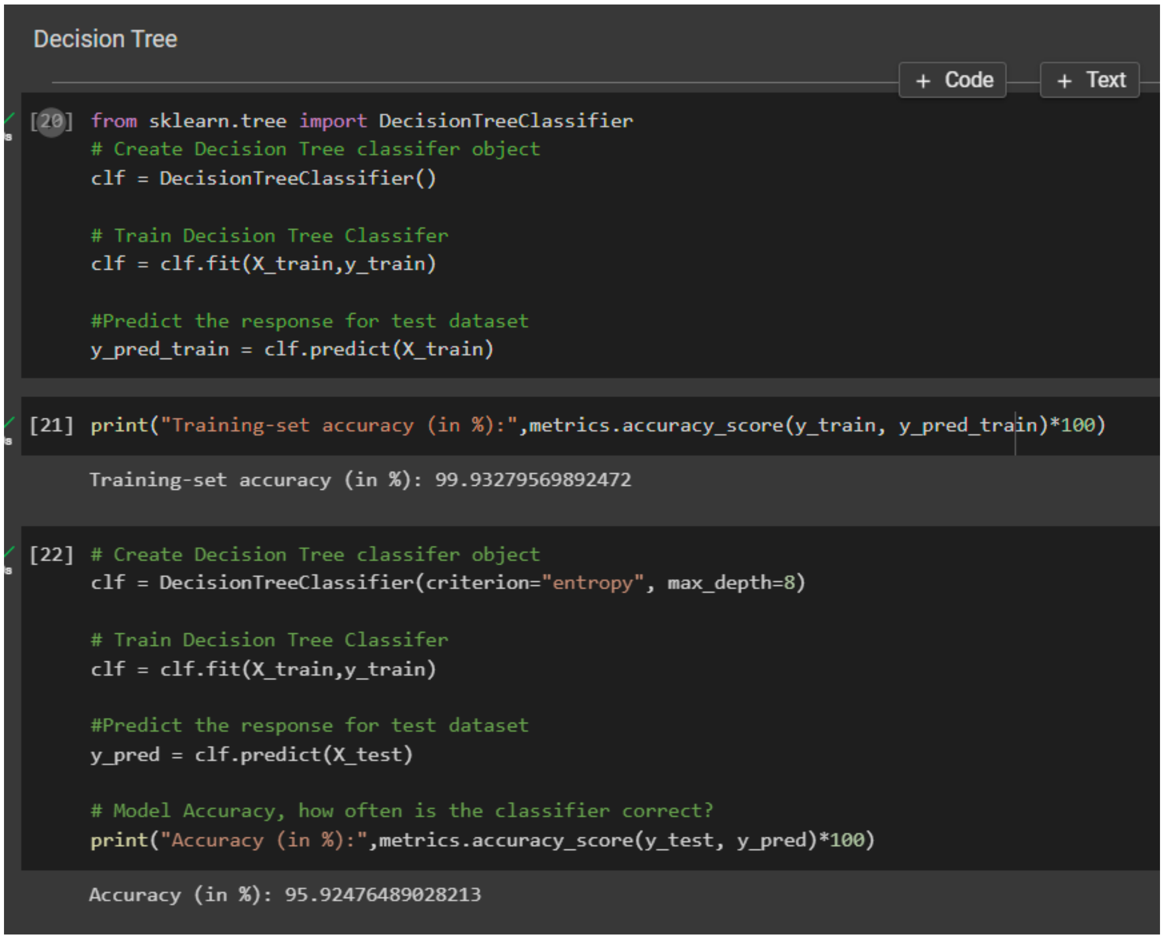
```
[28] #Import svm model
from sklearn import svm

#Create a svm Classifier
svmclf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
svmclf.fit(X_train, y_train)

y_pred_train = svmclf.predict(X_train)

print('Training-set accuracy(in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
Training-set accuracy(in %): 99.32795698924731
```

```
#Predict the response for test dataset
y_pred = svmclf.predict(X_test)

# using metrics module for accuracy calculation
print("SVM model accuracy(in %): ", metrics.accuracy_score(y_test, y_pred)*100)
```

```
SVM model accuracy(in %):  98.90282131661442
```

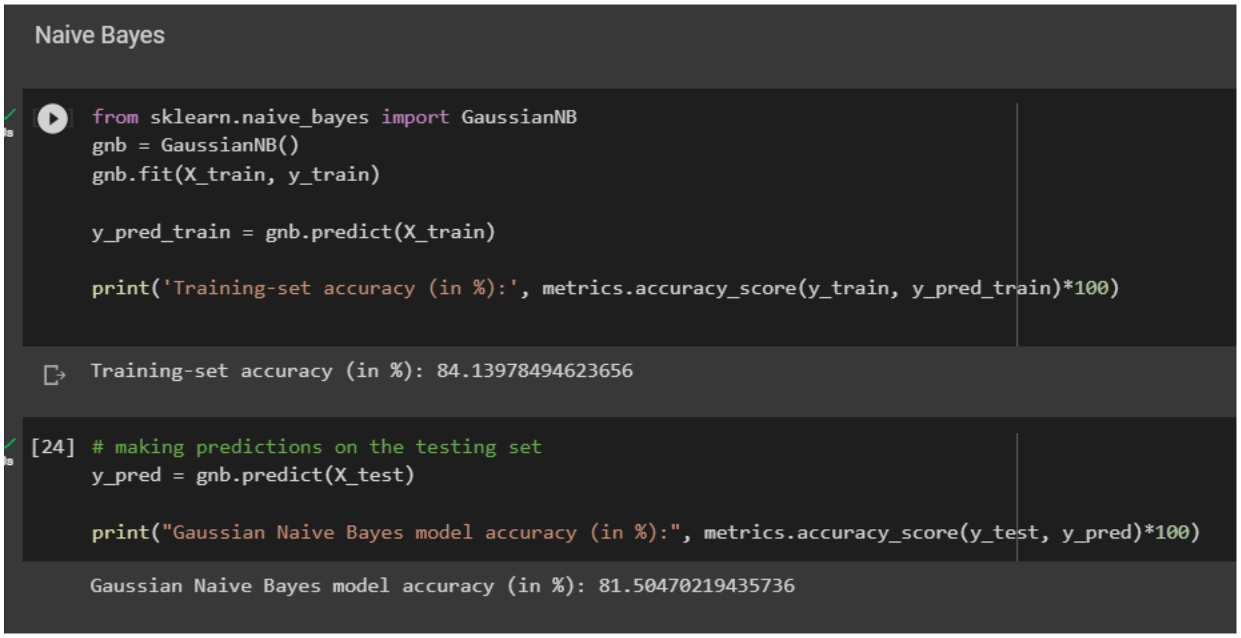
LDA:



```
[20] from sklearn.tree import DecisionTreeClassifier
     # Create Decision Tree classifer object
     clf = DecisionTreeClassifier()

     # Train Decision Tree Classifer
     clf = clf.fit(X_train,y_train)

     #Predict the response for test dataset
     y_pred_train = clf.predict(X_train)
```

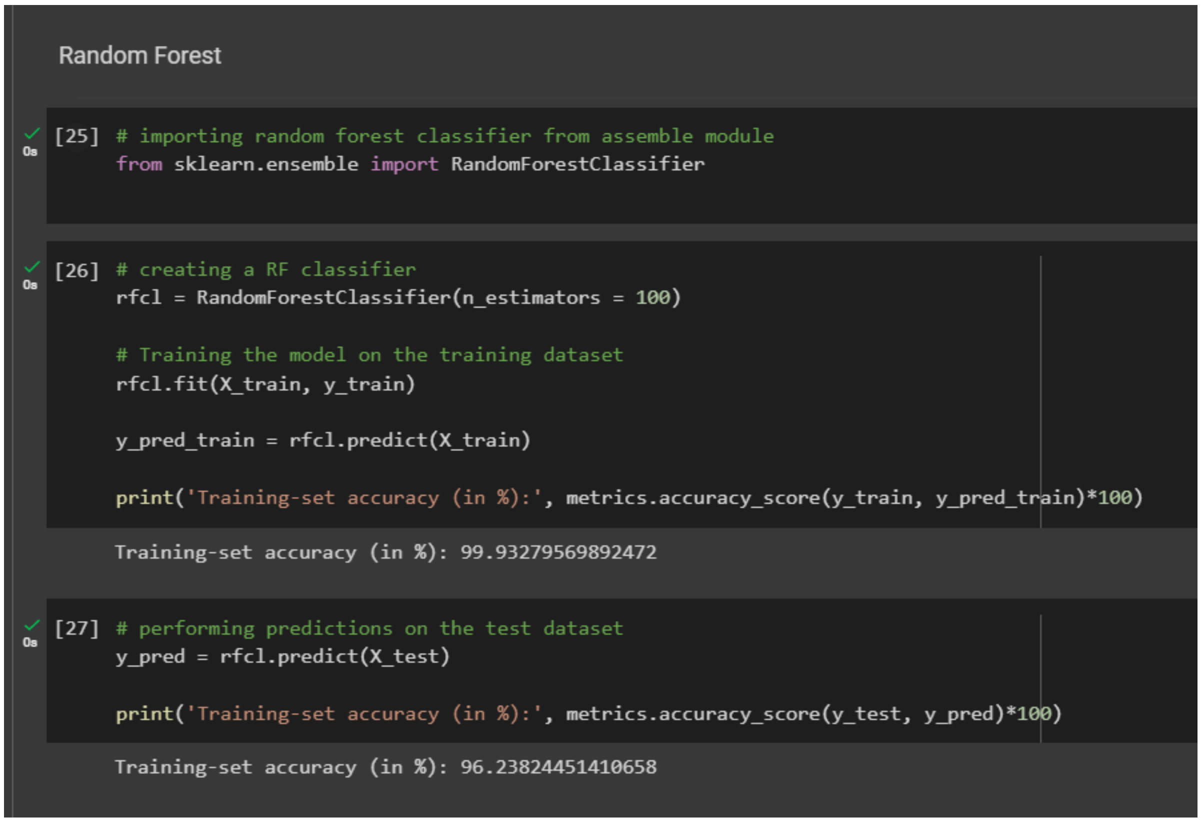
```
[21] print("Training-set accuracy (in %):",metrics.accuracy_score(y_train, y_pred_train)*100)

     Training-set accuracy (in %): 99.93279569892472
```

```
[22] # Create Decision Tree classifer object
     clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

     # Train Decision Tree Classifer
     clf = clf.fit(X_train,y_train)

     #Predict the response for test dataset
     y_pred = clf.predict(X_test)

     # Model Accuracy, how often is the classifier correct?
     print("Accuracy (in %):",metrics.accuracy_score(y_test, y_pred)*100)

     Accuracy (in %): 95.92476489028213
```



```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)

y_pred_train = gnb.predict(X_train)

print('Training-set accuracy (in %):', metrics.accuracy_score(y_train, y_pred_train)*100)

Training-set accuracy (in %): 84.13978494623656
```

```
[24] # making predictions on the testing set
     y_pred = gnb.predict(X_test)

     print("Gaussian Naive Bayes model accuracy (in %):", metrics.accuracy_score(y_test, y_pred)*100)

     Gaussian Naive Bayes model accuracy (in %): 81.50470219435736
```


Random Forest

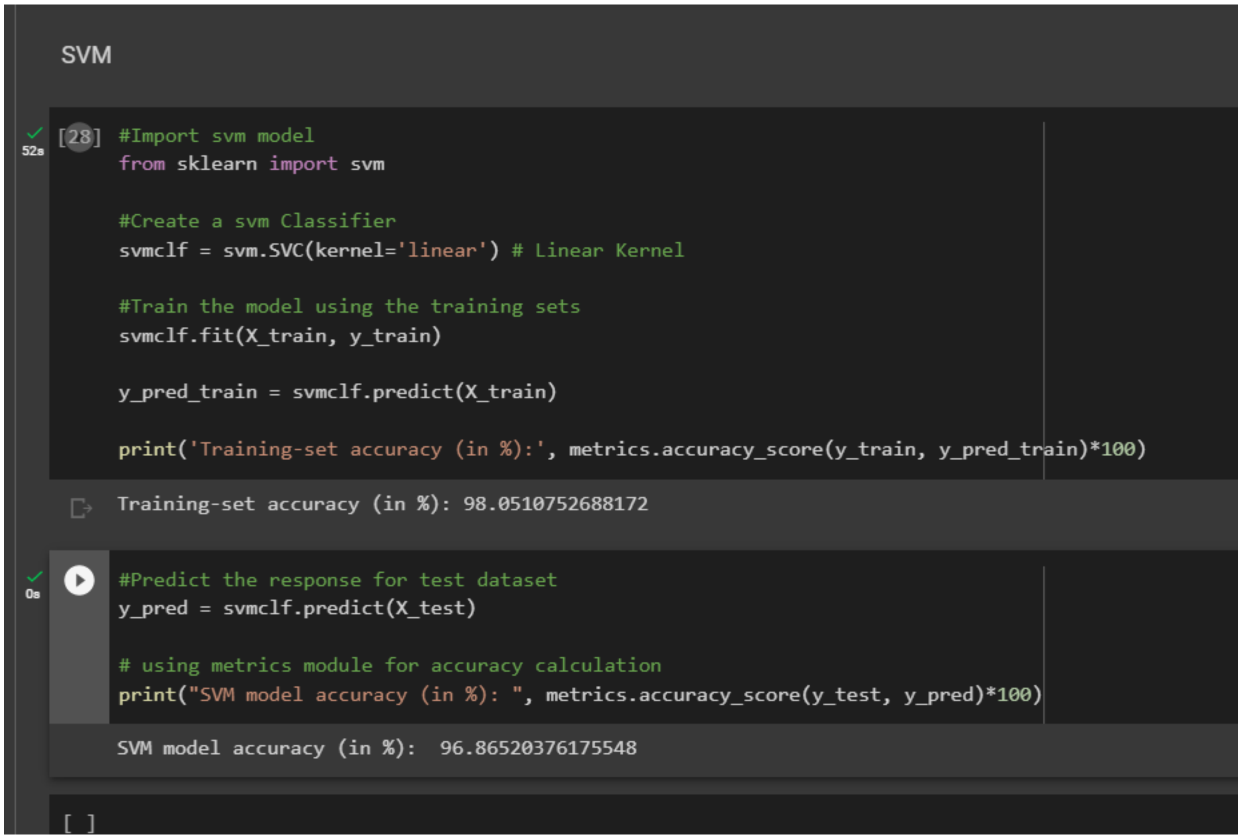
```
[25] # importing random forest classifier from assemble module
     from sklearn.ensemble import RandomForestClassifier
```

```
[26] # creating a RF classifier
     rfcl = RandomForestClassifier(n_estimators = 100)

     # Training the model on the training dataset
     rfcl.fit(X_train, y_train)

     y_pred_train = rfcl.predict(X_train)

     print('Training-set accuracy (in %):', metrics.accuracy_score(y_train, y_pred_train)*100)
```

```
     Training-set accuracy (in %): 99.93279569892472
```

```
[27] # performing predictions on the test dataset
     y_pred = rfcl.predict(X_test)

     print('Training-set accuracy (in %):', metrics.accuracy_score(y_test, y_pred)*100)
```

```
     Training-set accuracy (in %): 96.23824451410658
```



SVM

```
[28] #Import svm model
     from sklearn import svm

     #Create a svm Classifier
     svmclf = svm.SVC(kernel='linear') # Linear Kernel

     #Train the model using the training sets
     svmclf.fit(X_train, y_train)

     y_pred_train = svmclf.predict(X_train)

     print('Training-set accuracy (in %):', metrics.accuracy_score(y_train, y_pred_train)*100)

     Training-set accuracy (in %): 98.0510752688172
```

```
     #Predict the response for test dataset
     y_pred = svmclf.predict(X_test)

     # using metrics module for accuracy calculation
     print("SVM model accuracy (in %): ", metrics.accuracy_score(y_test, y_pred)*100)

     SVM model accuracy (in %):  96.86520376175548
```

FUTURE WORK:

While PCA and LDA are widely used techniques, other methods such as t-SNE, UMAP, and autoencoders could be investigated for their effectiveness on the given dataset.

Evaluating the influence of various hyperparameters: Hyperparameters like the number of components or the learning rate can have a substantial impact on the performance of dimensionality reduction approaches. Future work could include investigating the effect of various hyperparameters and determining the best values.

Using the same dimensionality reduction techniques on multiple datasets: The study might be expanded by using the same dimensionality reduction approaches on different datasets and comparing the results. This might offer insights on the strategies' efficacy on various sorts of data.

REFERENCES:

[1] W. Siblini, P. Kuntz, and F. Meyer, “A Review on Dimensionality Reduction for Multi-label Classification,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019, doi: https://doi.org/10.1109/tkde.2019.2940014.

[2] W. Zhao and S. Du, "Spectral–Spatial Feature Extraction for Hyperspectral Image Classification: A Dimension Reduction and Deep Learning Approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, Aug. 2016, doi: https://doi.org/10.1109/TGRS.2016.2543748.

[3] S. Feng and H. Wang, "Comparison of PCA and LDA Dimensionality Reduction Algorithms based on Wine Dataset," *IEEE Xplore*, May 01, 2021. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9602325&isnumber=9601344.
[4]
S. Poudyal, M. J. Mohammadi-Aragh, and J. E. Ball, "Data Mining Approach for Determining Student Attention Pattern," *IEEE Xplore*, Oct. 01, 2020. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9274061&isnumber=9273811).

[5]
N. Manohar, Y. H. Sharath Kumar, and G. H. Kumar, "Supervised and unsupervised learning in animal classification," *IEEE Xplore*, Sep. 01, 2016. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7732040&isnumber=7732013.

Code and Report:

https://github.com/SwethaKanakavalli/Bigdata.git