Hannah Kareti & Marianne Yamazaki Dorr
COEN 140
December 9, 2022

Final Project Report

Introduction

For our project, we decided to measure different classification techniques and observe which ones give us accurate data to predict whether someone has diabetes. By using the techniques and models that we were taught in class, we hope to find a method that can best predict diabetes outcomes by using a variety of metrics such as accuracy, precision, recall, F1 score, and so on. The dataset that we chose is from the National Institute of Diabetes and Digestive and Kidney Diseases. This data predicts whether a patient has diabetes and is based. This data consists of features including previous pregnancies, glucose, blood pressure , skin thickness, insulin, BMI, DiabetesPedigreeFunction, age, and class. The target variable for this data is the outcome, zero means having no diabetes and one means having diabetes. All the patients of this data are females that are twenty-one years of age.

Here is a brief look at the beginning of the data and its structure:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Our motivation for this project was our interest in the rapid growth of diabetes. According to the World Health Association, about 422 million people worldwide have diabetes. Diabetes is affecting a large population across the globe and is more prominent in individuals with compromised attributes such as age, gender and blood pressure.

We originally were planning to use another diabetes dataset focused on hospitalization outcomes which had over 100k entries. However, the data itself was difficult to deal with, requiring extensive pre-processing such as managing missing entries, converting non-numerical data into a numerical format that the machine learning models could understand, and identifying which features were relevant or not among the 50+ features listed, many of which had the mostly the same value for all entries. In order to get the project done in a timely manner, we chose this new dataset since:

1. The dataset was easy to understand and work with
2. Had a lower amount of features (allowing for easier feature selection)

The main downside with this dataset was its small number of entries, of around 700-800 entries. The other caveat was that we were now focusing on the presence/absence of diabetes rather than the hospitalization outcomes of diabetes, which could be more interesting to study in the future.

Methodology

We decided to use several different methods: QDA, LDA, Logistic Regression , Deep Learning, and Naive Bayes to measure our data. We performed accuracy tests with each of these models in order to determine which one would be best at predicting the presence or absence of diabetes based on the provided features.

LDA and QDA:

Both LDA and QDA can be derived from probabilistic models which model the class conditional distribution of the data $P(X|y=k)$ for each class k. LDA, short for Linear Discriminant analysis, is a linear classification method while QDA, short for Quadratic Discriminant Analysis, is a non-linear classification method, specifically it is quadratic. In order

to make predictions of which category a new entry belongs to, Bayes' rule is used for each training sample $x \in \mathcal{R}^d$:

For LDA and QDA, we used an 80/20 split for training and testing data, using the first 80% of the data for training the models and the last 20% for testing the models.

Mathematical equation for QDA:

$$\begin{aligned}\log P(y=k|x) &= \log P(x|y=k) + \log P(y=k) + Cst \\ &= -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^t\Sigma_k^{-1}(x-\mu_k) + \log P(y=k) + Cst,\end{aligned}$$

Mathematical equation for LDA:

$$\log P(y=k|x) = -\frac{1}{2}(x-\mu_k)^t\Sigma^{-1}(x-\mu_k) + \log P(y=k) + Cst.$$

Logistic regression

Logistic regression is a supervised classifier with a linear model to predict a binary outcome. It uses linear regression to predict an output. However since that output is not bounded it must be processed by a sigmoid function which will scale it from (-∞,∞) to (0, 1). If the outcome is closer to 0, then the new entry is predicted to belong to category 0 and if the outcome is closer to 1, then the new entry is predicted to belong to category 1. It works particularly well on simpler datasets but may be lacking on non-linear datasets.

$$h\Theta(x) = 1/1 + e - (\beta o + \beta 1X)$$

Deep Learning

Deep learning uses multiple layers of neurons which creates a neural network to learn complex patterns and relations within the data. Using non-linear activation functions between layers, the neural network is able to extract useful features and make predictions on the outcome of diabetes after being trained. We elected to use a 6 layer neural network. Each hidden layer used a reLU activation function and the output layer used a sigmoid function. We decided to run it through 500 epochs using a batch size of 50. We used the Keras library to model our neural network.

Naive Bayes:

Naive Bayes is a supervised classification model. It uses the Bayes theorem to calculate the probability that a new entry belongs to each category based on the features provided, in this case diabetes vs no diabetes. The category with the highest probability will then be predicted as the output for that new entry.

Libaries used:

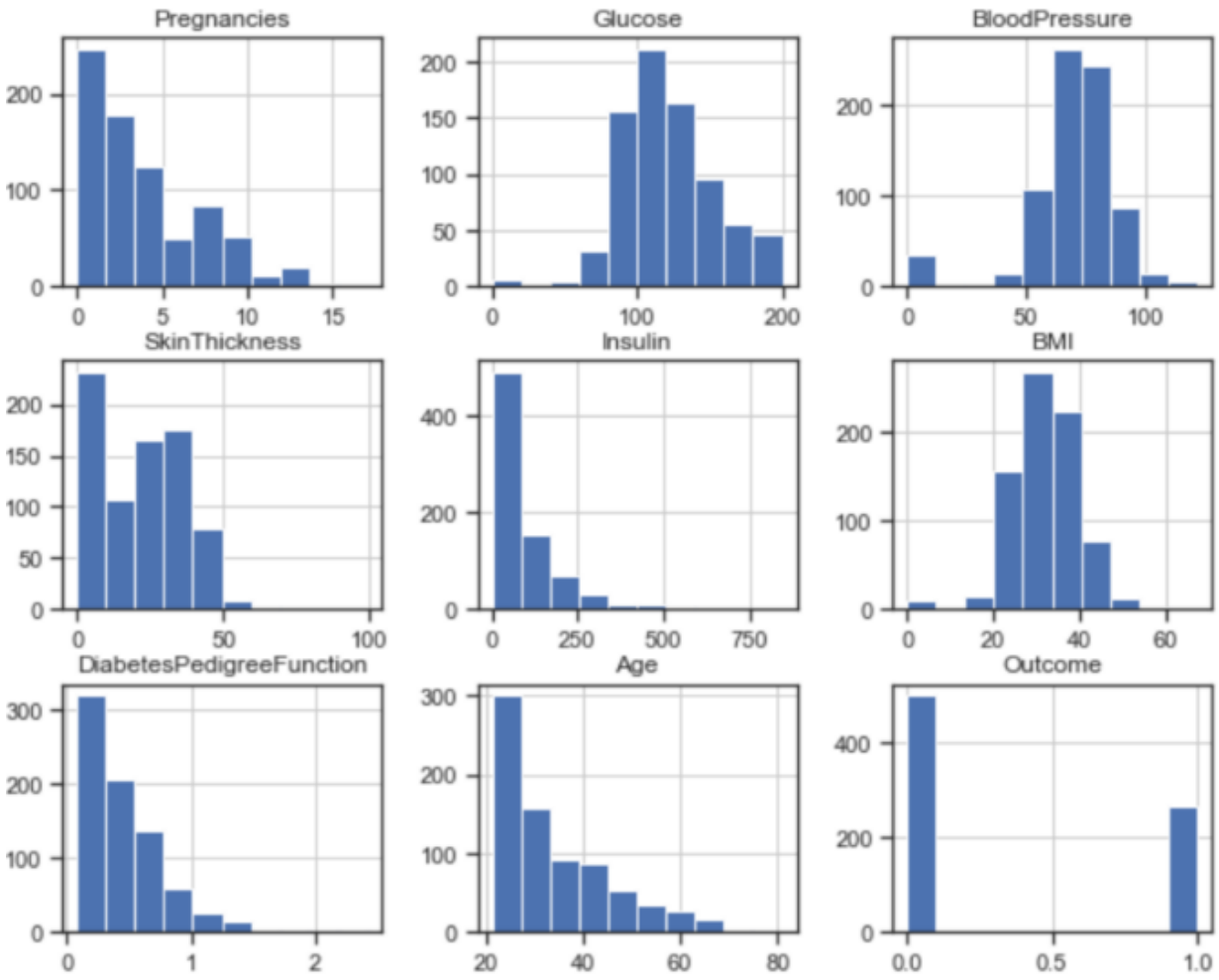
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import pandas as pd
```

Experimenting on our data:

Preprocessing our Data:

Our data consisted of 768 rows and 9 columns. We decided to split our data into training and testing with a ratio of eighty percent training data and twenty percent testing data. In addition, we made the y variable as the target variable (diabetes outcome) leaving x to be the rest of the features.

Initially, we performed some brief data analysis on the distribution of the features within the data set, just to get an initial picture of the type of data we may be dealing with. As a result, we graphed these histograms of each feature:



We also plotted the scatterplots of each feature vs every other feature,where the more scattered the data, the less correlated the features are. The more linear the data, the more correlated the features are. :

Results & Analysis :

Using LDA and QDA

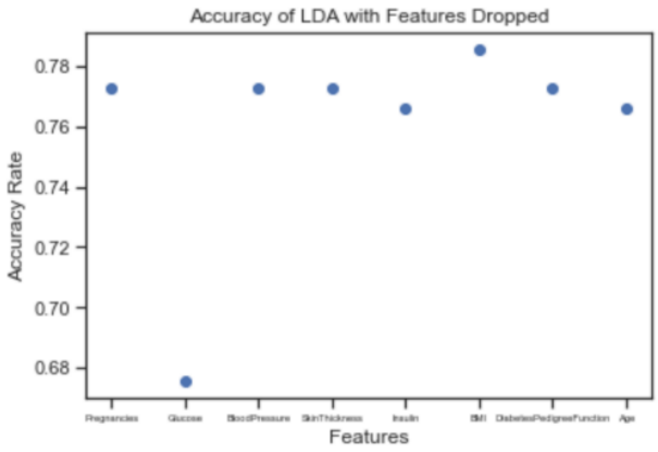
```
lda = LinearDiscriminantAnalysis(solver="svd", store_covariance=True)
lda = lda.fit(x_train, y_train)
y_pred_training = lda.predict(x_train)
y_pred_test = lda.predict(x_test)

#Accuracy on Training Data Set
score = lda.score(x_train, y_train)
print("LDA score on training", score)

#Accuracy on Testing Data Set
score = lda.score(x_test, y_test)
print("LDA score on testing", score)

#QDA
qda = QuadraticDiscriminantAnalysis(store_covariance=True)
qda = qda.fit(x_train, y_train)
y_pred_training = qda.predict(x_train)
y_pred_test = qda.predict(x_test)

#Accuracy on Training Data Set
score = qda.score(x_train, y_train)
print("QDA score on training", score)

#Accuracy on Testing Data Set
score = qda.score(x_test, y_test)
print("QDA score on testing", score)
```

We then used the LDA and QDA libraries to measure the accuracy of the testing and training data. We observed that both the accuracy in LDA and QDA are similar falling around 70-80%.

```
LDA score on training 0.7801302931596091
LDA score on testing 0.7727272727272727
QDA score on training 0.7703583061889251
QDA score on testing 0.7207792207792207
```

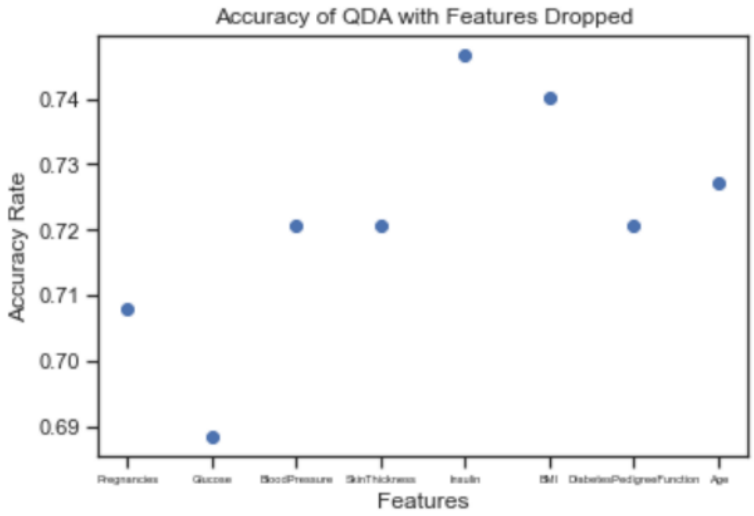
LDA Feature Dropping

As we visualized the data, we observed that as accuracy goes down most drastically when the Glucose feature is removed. This indicates that this feature is strongly correlated to the diagnosis of Diabetes.



QDA Feature Dropping

Similarly to LDA, the model becomes most inaccurate when the Glucose feature is removed. Thus reinforcing its position as the most important feature. On the other hand, model accuracy goes up when the Insulin feature is removed, thus making it the least important feature.



Logistic Regression

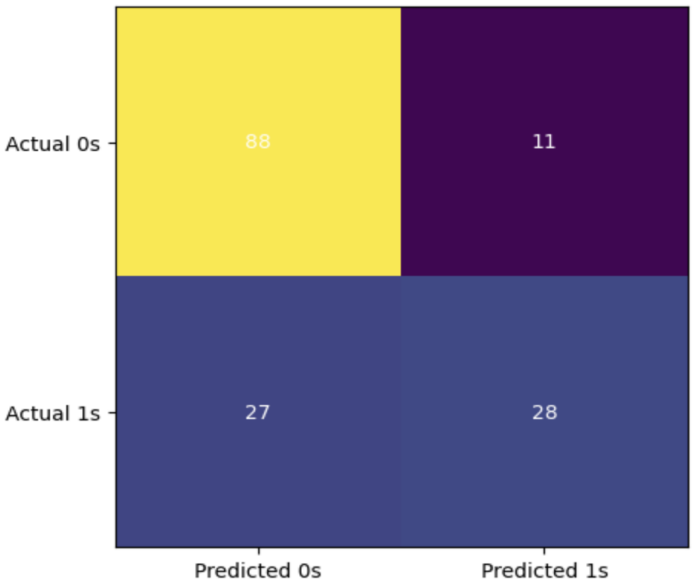
With our trained data we also measured the accuracy using logistic regression. We found that logistic regression produces similar results predicting the accuracy to be around 75%.

```
#Logistic Regression
classifer = LogisticRegression()
#fit the model using the training data
classifer.fit(x_train,y_train)
#use model to make predictions on test data
y_pred = classifer.predict(x_test)
```

```
Accuracy for Logistic Regression: 0.7532467532467533
```

Because logistic regression is a binary classification model, it is also best to produce a confusion matrix which will portray the true negatives, false negatives, false positives, and true positives.

```
heatmap = confusion_matrix(y_test,y_pred)

fig, ax = plt.subplots(figsize=(5, 5))
ax.imshow(cm)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))
ax.set_ylim(1.5, -0.5)
for i in range(2):
    for j in range(2):
        ax.text(j, i, cm[i, j], ha='center', va='center', color='white')
plt.show()
```



The obtained matrix shows the following:

- **Eighty eight true negative predictions:** The first three observations are zeros predicted correctly.
- **Twenty seven false negative predictions:** These are the ones wrongly predicted as zeros
- **Eleven false positive predictions:** The fourth observation is a zero that was wrongly predicted as one.
- **Twenty eight true positive predictions:** The last six observations are ones predicted correctly

In addition, we also measured the accuracy by printing out the classification report of our given dataset. The classification report is one of the performance evaluations that displays the

precision, recall, F1 score, and the support. Each metrics shows us the following:

Precision: Precision is the ratio of true positives to the sum of true and false positives.

Recall: Recall is the ratio of true positives to the sum of true positives and false negatives.

F1 score: The F1 is the weighted harmonic mean of precision and recall. The closer the value of the F1 score is to 1.0, the better the expected performance of the model is.

Support: Support is the number of actual occurrences of the class in the dataset. It doesn't vary between models, it diagnoses the performance evaluation process.

```
In [59]: print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.77      0.89      0.82        99
               1       0.72      0.51      0.60        55

        accuracy                           0.75       154
       macro avg       0.74      0.70      0.71       154
    weighted avg       0.75      0.75      0.74       154
```

<u>Naive Bayes:</u>

We used Naive Bayes to also measure accuracy because we know that through naive bayes, the bayes theorem is used giving us strong independent assumptions between the features. They can produce high accuracy levels and are highly scalable, requiring all the parameters to be linear.

```
#Calling the Class
naive_bayes = GaussianNB()

#Fitting the data to the classifier
naive_bayes.fit(x_train , y_train)

#Predict on test data
y_predicted = naive_bayes.predict(x_test)
metrics.accuracy_score(y_predicted , y_test)
```

```
0.7467532467532467
```

Deep Learning:

Layer (type)	Output Shape	Param #
dense_43 (Dense)	(None, 20)	180
dense_44 (Dense)	(None, 15)	315
dense_45 (Dense)	(None, 10)	160
dense_46 (Dense)	(None, 12)	132
dense_47 (Dense)	(None, 8)	104
dense_48 (Dense)	(None, 1)	9

Total params: 900
Trainable params: 900
Non-trainable params: 0

Accuracy among all the Models

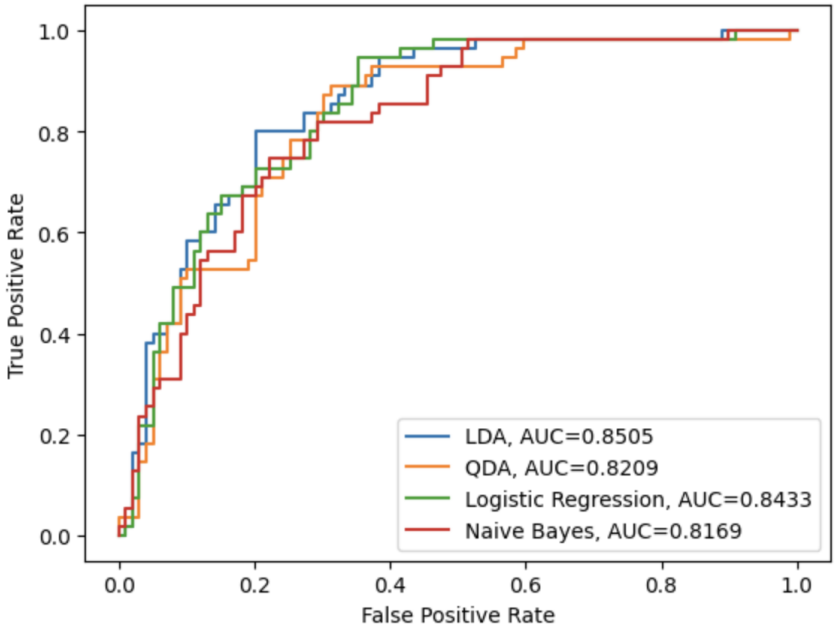
We wanted to measure the accuracy among all the classification models with the exception of our deep learning neural network. We decided to use the ROC curve to produce a line graph that we could visualize which classification model produces the best results.

ROC:

The ROC curve stands for a receiver operator characteristic curve. This curve is a graphical representation of the diagnostic ability of binary classifiers. The ROC curve plots the true positive rates against the false positive rate. The best measurement to interpret the ROC curve is to see the performance of each classifier into a single measure. The AOC, which is the

area under the curve, measures the probability that a randomly chosen positive instance is ranked higher than a random negative instance. The higher the AOC, the more accurate the model is.

```
model = LinearDiscriminantAnalysis(solver="svd", store_covariance=True)
model.fit(x_train, y_train)
y_pred = model.predict_proba(x_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
auc = round(metrics.roc_auc_score(y_test, y_pred), 4)
plt.plot(fpr,tpr,label="LDA, AUC="+str(auc))

model = QuadraticDiscriminantAnalysis(store_covariance=True)
model.fit(x_train, y_train)
y_pred = model.predict_proba(x_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
auc = round(metrics.roc_auc_score(y_test, y_pred), 4)
plt.plot(fpr,tpr,label="QDA, AUC="+str(auc))

#fit logistic regression model and plot ROC curve
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict_proba(x_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
auc = round(metrics.roc_auc_score(y_test, y_pred), 4)
plt.plot(fpr,tpr,label="Logistic Regression, AUC="+str(auc))

#fit gradient boosted model and plot ROC curve
model = GaussianNB()
model.fit(x_train, y_train)
y_pred = model.predict_proba(x_test)[:, 1]
fpr, tpr, _ = metrics.roc_curve(y_test, y_pred)
auc = round(metrics.roc_auc_score(y_test, y_pred), 4)
plt.plot(fpr,tpr,label="Naive Bayes, AUC="+str(auc))
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')

#add legend
plt.legend()
```



As we can see from the ROC curve, the LDA produced the best AOC, so that classification measurement has a better accuracy than others.

Conclusion and future work

In conclusion, out of all the models we have used and tested, we received the best results with LDA, achieving the highest prediction accuracy of 77.2%. Our worst performing model was our Deep Learning neural network model which only achieved a 73.38% prediction accuracy. Additionally, using feature dropping for both LDA and QDA, we concluded that the most significant feature was glucose, since when it was removed, the accuracy of both models tanked. Conversely, the least important feature was insulin. With its removal, both LDA and QDA models actually improved in accuracy.

For future work, it would be ideal to work on a larger dataset, such as the one we originally intended to use and to also go back to predicting hospitalizations outcomes instead of simply predicting diabetes outcomes, since gaining more data on how to treat and understand the

impact of diabetes is a much more complicated problem than testing the presence of diabetes and thus would be more beneficial to conduct more research on. Additionally, we would like to use different types of models that we did not get a chance to use in this project such as Decision Trees, Random Forests, SVMs (support vector machines), and to further experiment with neural networks by tweaking the hyperparameters and see if we can solve our overfitting problem.