Udacity Machine Learning Capstone Project- Aug 2020

Diabetes Prediction with Sagemaker

Amudha Narayanan

1. Definition

1.1 Project Overview:

Today world problem is COVID -19, which have 7.3% fatality rate on patients with Diabetes comparing to other patients is 2.3 % [1]. Many complications occur if diabetes remains untreated and unidentified in COVID situation as diabetes is one of the deadliest and chronic diseases which causes an increase in blood sugar end up in multiple organism failure with COVID. With the current situation it is not recommended to visit diagnostic center and hospital multiple times just for diagnosing. Machine learning approaches solves this critical problem at the early stage by analyzing the predicted result with online doctor consultation. As the vaccination for COVID-19 is not still available early detection of diabetes help patients and world health sector to focus on health and infrastructure planning in advance to handle growing number of diabetes patients which is forecasted to be 642 million by 2040.

As my parents have diabetes and as a Machine Learning Engineer working in Research and Technology, I always have intention to build Diabetes predictor, now Udacity ML nanodegree course had now given me confident to explore diabetes dataset and use AWS Sagemaker to come with end to end solution for Diabetes Predictor.

The Prima Indian Diabetes Dataset with 768 patient's data has been used in this study, provided by the UCI Machine Learning Repository (https://www.kaggle.com/uciml/pima-indians-diabetes-database). The dataset has been originally collected from the National Institute of Diabetes and Digestive and Kidney Diseases. The number of true cases are 268 (34.90%) and the number of false cases are 500 (65.10%) in this dataset.

1.2 Problem Statement:

The motive of this study is to design a model which can prognosticate the likelihood of diabetes in patients with maximum accuracy just by invoking Endpoint of Sagemaker with parameters they got from the Glucose Tolerance Test(GTT) result done recently before COVID or from laboratory results done in remote. The model result will be of binary classification The results of this model can then be used to help detect diabetes in an earlier stage to allow individuals to live a healthier lifestyle and be alert in protecting themselves from coronavirus until vaccination is found and safe to inject to all.

The problem of diagnosing diabetes mellitus is a binary classification problem; therefore, it can be handled by classification techniques with more than one Machine Learning algorithms such as Logistic Regression, SVC, XGBoost classifier to predict whether a patient have diabetes or no diabetes based on the measurements captured with Glucose Tolerance Test(GTT).

1.3 Metrics

The model is primarily evaluated based on overall accuracy, since the real dataset doesn't have equal samples belong to each class, it is always good to evaluate with more than one classification metrics as it important to consider the percentage of false positives and false negatives. Since our model predicts a disease state, incorrectly predicting diabetes for a patient could be upsetting to the patient and lead to unnecessary actions or tests while incorrectly missing a diagnosis could lead to a patient developing the very complications which defeat goal of model. So evaluated model by

1. Calculating classification **Accuracy** where Accuracy is the ratio of number of correct predictions to the total number of input samples.
2. Plotting Receiver Operating Characteristic(**ROC**) between True Positive Rate and False Positive Rate, the larger the Area Under Curve(**AUC**), the better is the model
3. Calculating **F1 score** which is the Harmonic Mean between precision and recall, greater the F1 Score, the better is the performance of the model.The F1 score is the harmonic mean of precision and recall. It assigns equal weight to both the metrics. However, for our analysis it is relatively more important for the model to have low false negative cases (as it will be dangerous to classify high risk patients in low risk category). Therefore, we individually look at Precision and Recall.

2. Analysis

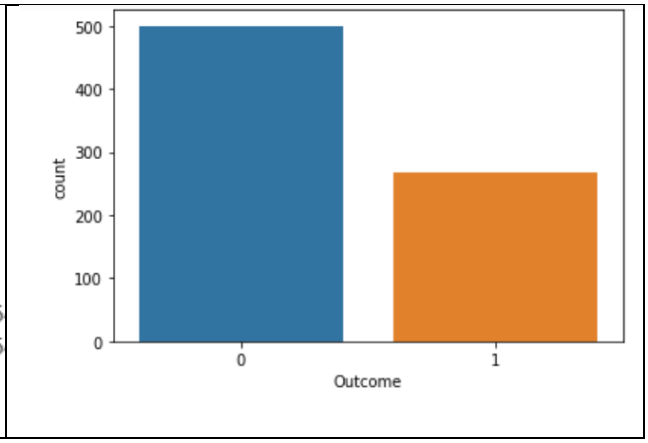
2.1 Data Exploration

The Prima Indian Diabetes Dataset have 768 observations with following 8 medical predictor features

Features	Description	Data type
Pregnancies	Number of times pregnant	int64
Glucose	Plasma glucose concentration a 2 hour in GTT	int64
BloodPressure	Diastolic blood pressure (mm Hg)	int64
SkinThickness	Triceps skin fold thickness (mm)	int64
Insulin	Hour serum insulin (mu U/ml)	int64
BMI	Body mass index (weight in kg/(height in m)^2)	Float64
DiabetesPedigreeFunction	Likelihood score of diabetes based on family history	Float64
Age	Age of Patient (years)	int64

The Prima Indian diabetes dataset have one target variable 'outcome' (output 0 for "no diabetes" or 1 for "diabetes"). Let's check the feature and target variable distribution from data exploratory study:

```
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float6
 6   DiabetesPedigreeFunction  768 non-null    float6
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
```



Observations:

- There are a total of 768 records and 9 columns with 8 medical feature and 1 target outcome in the dataset.
- Each feature was either of integer or float data type.
- In the outcome column, 1 represents diabetes positive and 0 represents diabetes negative
- There are 500 patients with diabetes negative and 268 with diabetes positive

Statistics summary of dataset

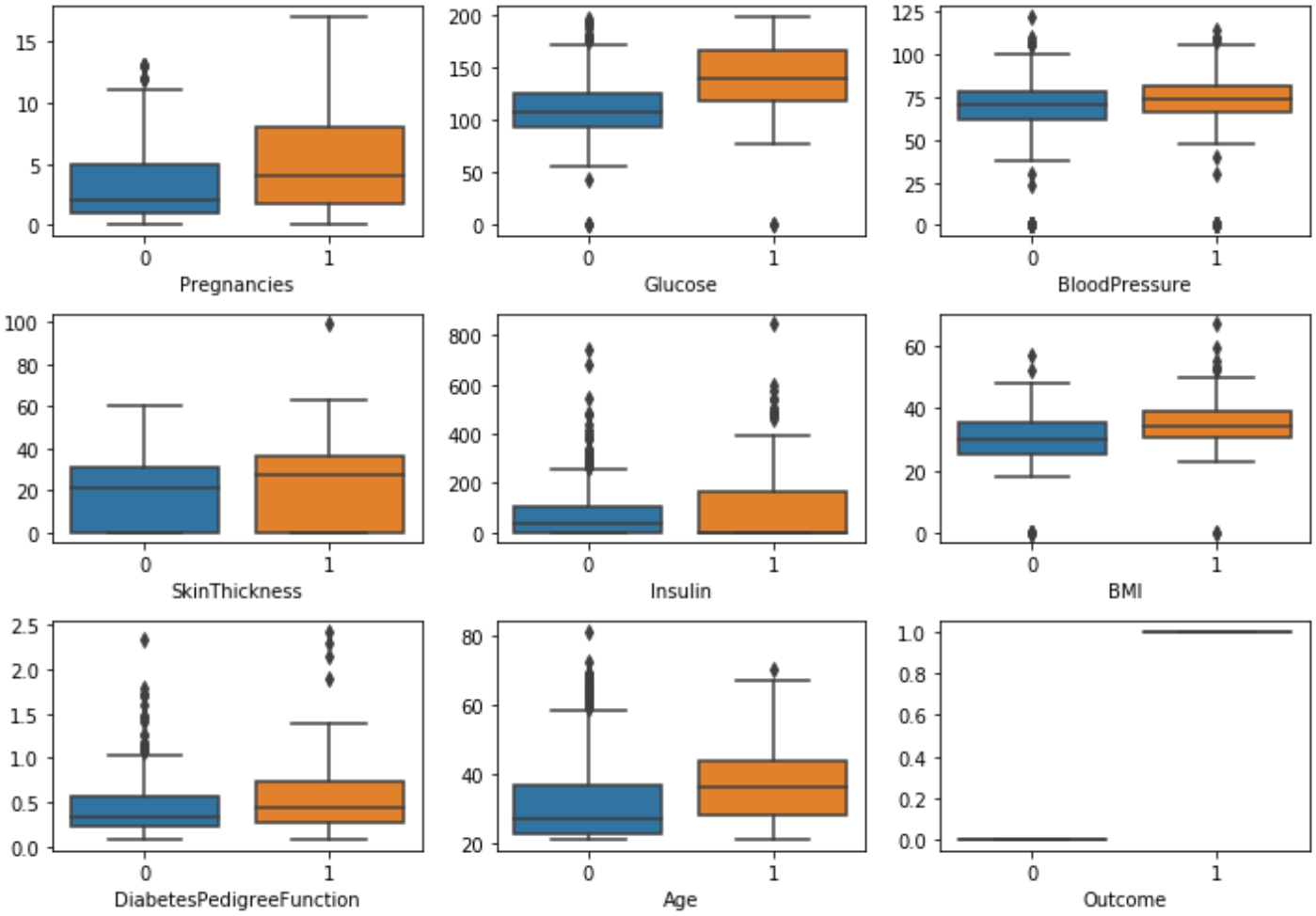
Given below table shows summary statistics such as mean/median and interquartile ranges for binary outcome target variables

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

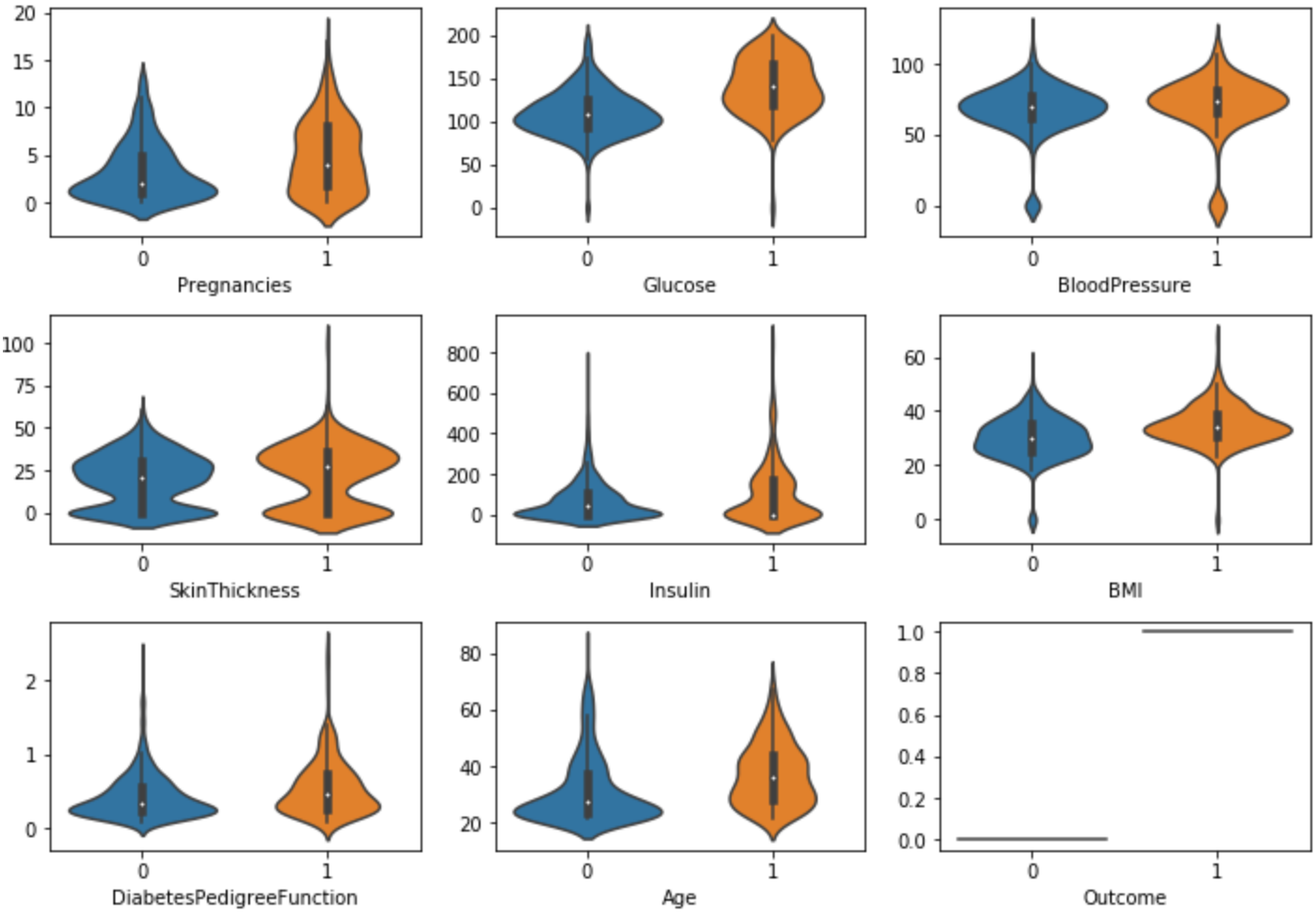
2.2 Exploratory Visualization

Above structured statistics detail of data can be represented through various types of plots. In that box plot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”). It can tell about outliers and what their values are. It can also tell if data is symmetrical, how tightly data is grouped,

and skewed. Given below box plot shows summary statistics such as mean/median and interquartile ranges for binary outcome target variables



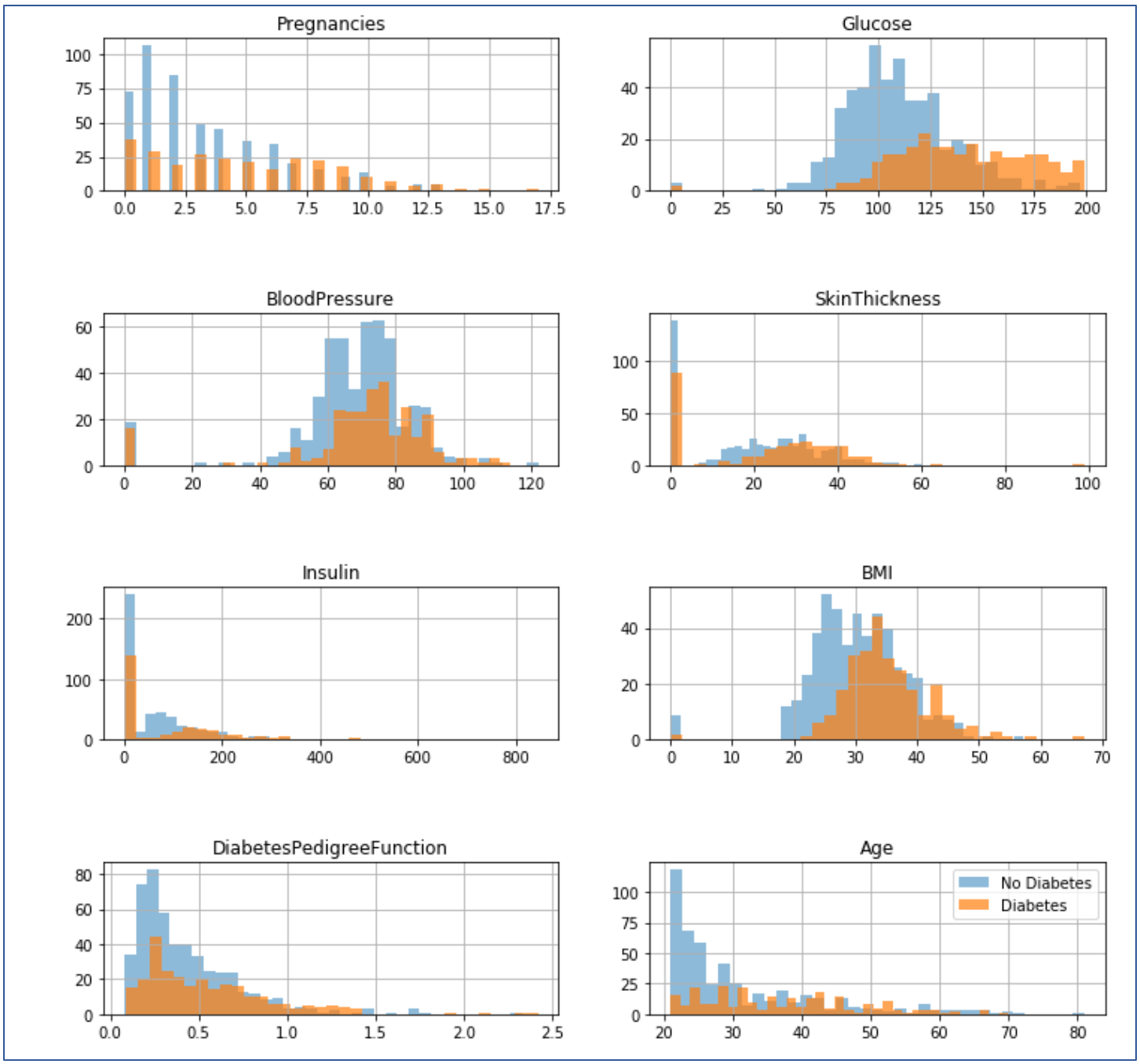
Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator. Typically, a violin plot will include all the data that is in a box plot: a marker for the median of the data; a box or marker indicating the interquartile range; and possibly all sample points as given below. The bottom tail of the *violins* indicates the zero values, the long tail at the top of the violin indicates the chances of having outlier in features



Observations:

- Analyzing *Glucose*, we observe the variable not following the normal distribution. We encounter zero-values in this instance as well. There are 5 such values for which treatment is required
- Observing the violin plot, we see a massive vertical distance between the box-plot for Diabetics and Non-Diabetics. This indicates that Glucose can be a very important variable in model-building
- The BMI variable seems to be closely following the normal distribution as the mean and median are approximately equal. Diabetic women had more pregnancies than non-diabetic. Skin Thickness for Diabetics is more than that of Non-Diabetics and there is one outlier
- Diabetes Pedigree Function is a positively skewed variable with no zero values.

To cross check that the above observation is correct we will continue further Exploratory Visualization with histogram



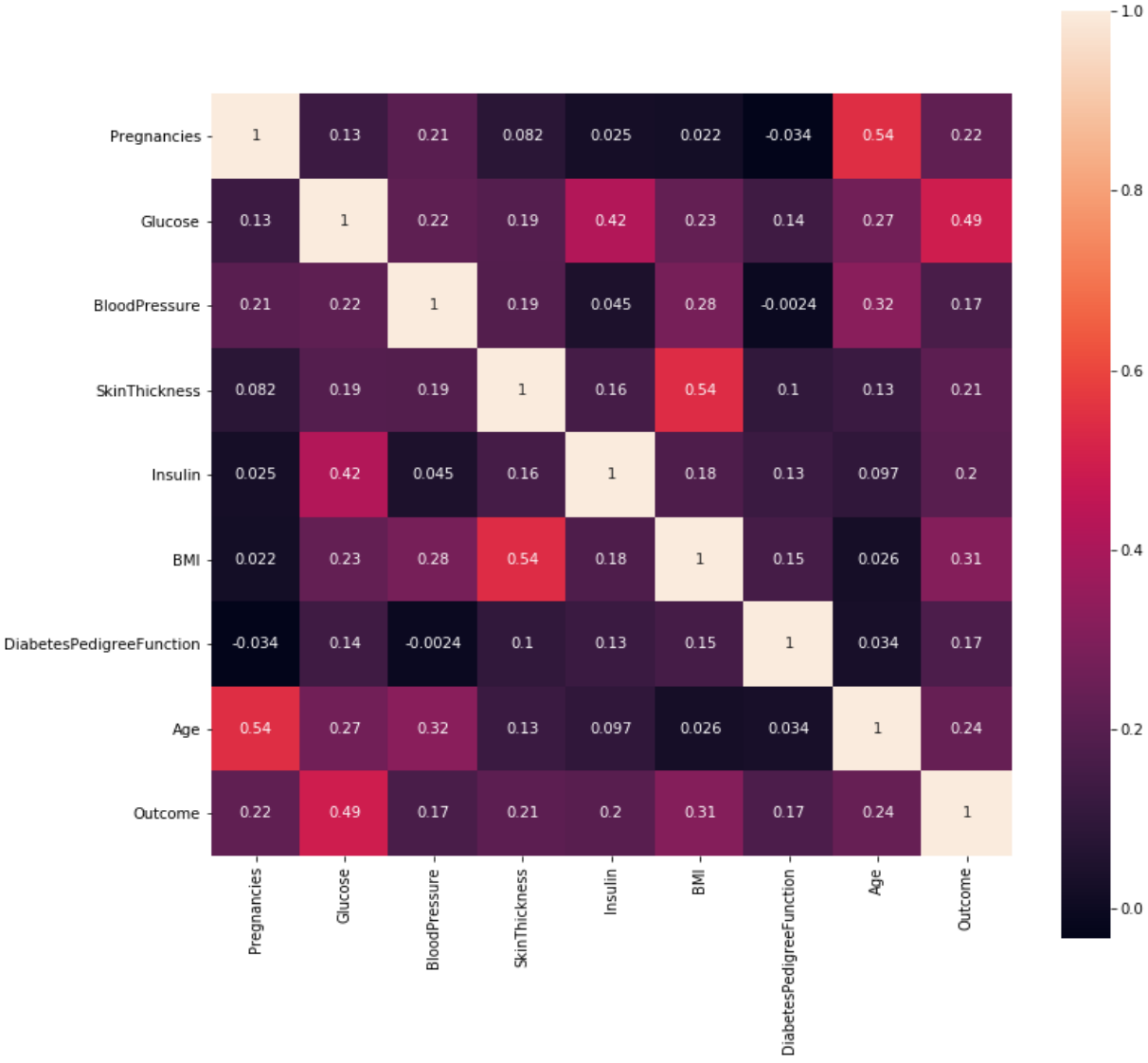
Visual inspection of all above plots of a summary of the data confirms that there are a number of features with minimum values as 0. Some features like Glucose, Blood pressure, Insulin, BMI have zero values which represent missing data. There are zero NaN values in the dataset.

Features:	**Total count of Zero Values**
Glucose:	5
Blood Pressure:	35
SkinThickness:	227
Insulin:	374
BMI:	11

Interpretation and justification of "0" value depends on which feature is being considered. In the case of "BloodPressure" for instance, a 0 must represent a missing value because an actual blood pressure value of 0 is not possible for any patient. On the other hand, in the case of "PregnanciesNumber", a 0 represents a valid observational value that a patient had 0 pregnancies.

Features	# of Zero values	Remarks
Pregnancies	111	Valid measurement , data pre-processing -replacing zero value with mean/median is not required
Glucose	5	Not a valid measurement, required to pre-process data to replace with median
BloodPressure	35	Unlikely to be a valid measurement, required to pre-process data to replace with median
SkinThickness	227	Unlikely to be a valid measurement, required to pre-process data to replace with median
Insulin	374	Unlikely to be a valid measurement, required to pre-process data to replace with median
BMI	11	Unlikely to be a valid measurement, required to pre-process data to replace with median
DiabetesPedigreeFunction	0	There is no patient with value of this as 0 , but 0 can be a valid measurement representing hereditary risk of diabetes
Age	0	There is no patient with age zero as the dataset include only female Patient with above age 20 years

Finally, a correlation matrix is used here to show how the features are closely related and to show their relationship with the outcome.



The correlation matrix above uses Pearson's correlation coefficient to illustrate the relationship between variables, there was a significant correlation can be observed between Pregnancies and Age. To further confirm, we calculate the correlation coefficient and found that other parameters Glucose, BMI and Age have good value of correlation co-efficient. By a rule of thumb, in case of correlation co-efficient above 0.70, multi-collinearity is expected. Hence, no significant case of multi-collinearity is observed. So I decided to build models with all features, but I implemented a study on Feature importance by using Random classifier for detail refer to Feature Importance section

2.3 Algorithms and Techniques

In capstone proposal, I proposed to build multiple supervised learning classifier model and at least one Scitkit learn model with AWS Sagemaker as the course made me confident to use Sagemaker for Machine learning. Now in my capstone project of Diabetes Predictor I had built multiple supervised learning models of Scitkit learn from linear classifier, tree classifier, naive bayes, neighbors classifier type as base models to compare with XGBoost models build with pre-built container image from AWS Sagemaker for training and deployment. My choice was to use the XGBoost algorithm based both on recommendations from proposal reviewers and its current widespread success across a variety of problems. XGBoost is also Scikit supervised learning model for which AWS SageMaker has a pre-built container image for training and deployment to simplify maintenance and deployment of the model.

The XGBoost classifier have several parameters that you can tune to improve the outcome:
• max_depth : int. Maximum tree depth for base learners.
• learning_rate : float. Boosting learning rate (xgb's "eta")
• n_estimators : int. Number of boosted trees to fit.
• silent : Boolean. Whether to print messages while running boosting.
• objective : string or callable. Specify the learning task and the corresponding learning objective
• booster: string. Specify which booster to use: gbtree, gblinear or dart.
• nthread : int. Number of parallel threads used to run xgboost.
• n_jobs : int. Number of parallel threads used to run xgboost.
• gamma : float. Minimum loss reduction required to make partition on a leaf node of the tree.
• min_child_weight : int. Minimum sum of instance weight(hessian) needed in a child.
• max_delta_step : int. Maximum delta step we allow each tree's weight estimation to be.
• subsample : float. Subsample ratio of the training instance.
• colsample_bytree : float. Subsample ratio of columns when constructing each tree.
• colsample_bylevel : float. Subsample ratio of columns for each split, in each level.
• reg_alpha : float (xgb's alpha). L1 regularization term on weights
• reg_lambda : float (xgb's lambda). L2 regularization term on weights
• scale_pos_weight : float. Balancing of positive and negative weights.
• base_score: The initial prediction score of all instances, global bias.
• seed : int. Random number seed. (Deprecated, please use random_state)
• random_state : int. Random number seed. (replaces seed)
• missing : float, optional. Value in the data which needs to be present as a missing value. If None, defaults to np.nan.

These parameters can be toned to improve the prediction accuracy, control overfitting and handle imbalanced data.

2.4 Benchmark

Many Kaggle projects with Prima Indian Diabetes dataset with 8 attributes able to get the accuracy for Logistic regression in average around 75%, In this SageMaker Diabetes Predictor project targeting to achieve accuracy around 73% to 76% with XGBoost classifier pre-built container of SageMaker by processing the Prima Indian Diabetes dataset and by hyper parameter tuning.

3. Methodology

3.1 Data Preprocessing

Applied the following pre-processing steps to diabetes dataset to build the baseline models with processed data

Remove outliers

From the Data visualization and Exploratory study found that there is one outlier in the "Skin thickness" we can remove it because it is most likely mistyped. For our Sagemaker XGBoost no need to remove outlier because "Decision family algorithm" like XGBoost can handle it.

Replace Zero with median

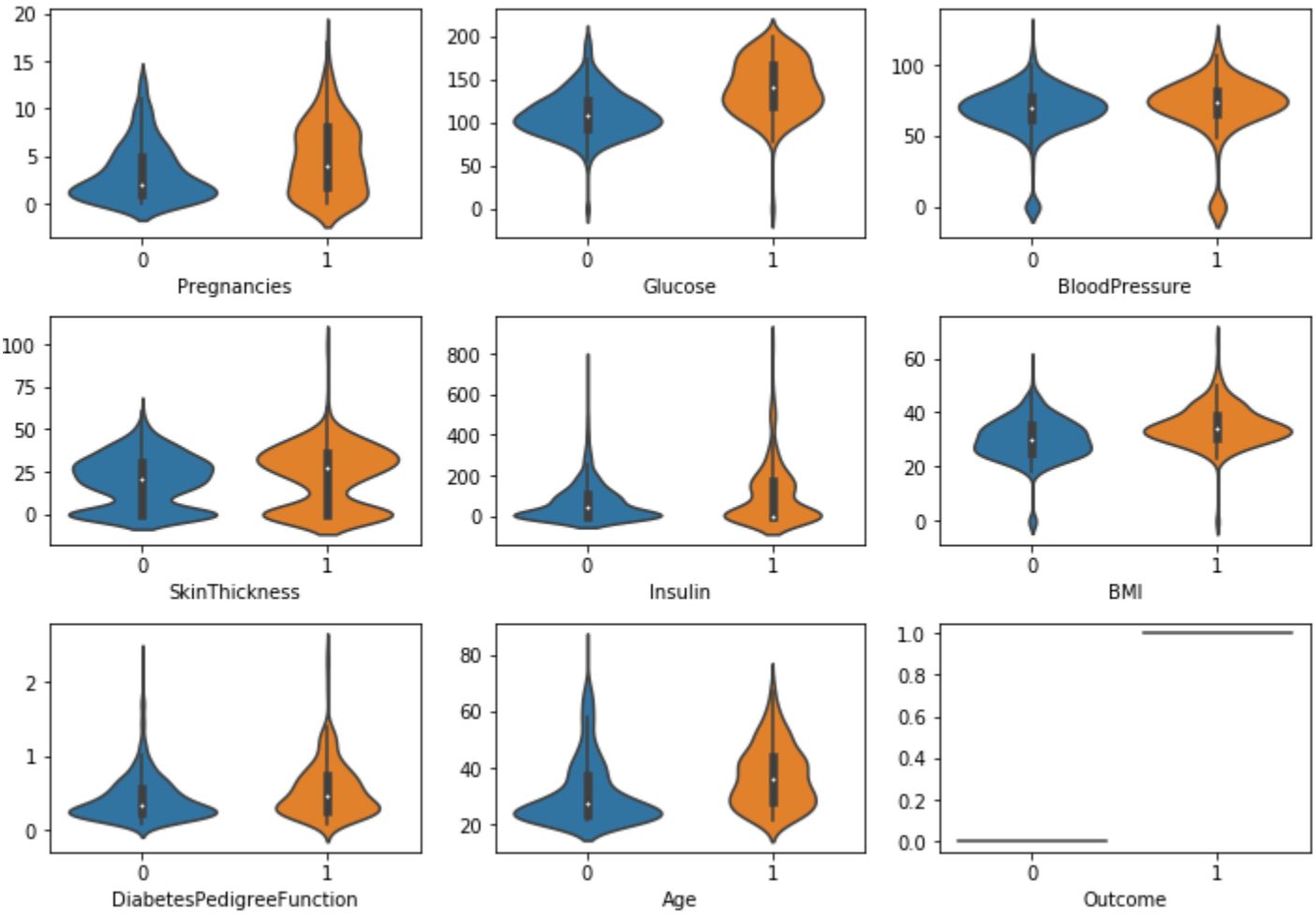
With Prima diabetes dataset features such as "Glucose","BMI", "Insulin", "Blood Pressure" and "Skin thickness" has zero values which is medically not possible for any patient. Removing zero value records is not recommended as the dataset is small and when we drop observations, we drop information and we will have less data to train and test, so for missing numeric data, we should flag and fill the values but this also leads to a loss in information but less comparing to dropping records. Dropping feature is not recommended as we have only 8 features and heat map also show they are co-related to outcome, in addition we can consider to drop feature if more than 60 % of observation is missing, but in diabetes dataset none of the feature meeting this criteria, feature ‘insulin’ alone have around 48 % zero value missing, so decided not to drop any feature.

Replacing with Mean/Median is recommended for diabetes dataset as the features Glucose","BMI", "Insulin", "Blood Pressure" and "Skin thickness" has numeric data We can calculate the mean, median or mode of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns.

Given below Violin plot and Statistical tabular report after pre-processing

Statistics summary of dataset after pre-processing

	count	mean	std	min	25%	50%	75%	max
Pregnancies	767.0	3.847458	3.371117	0.000	1.0000	3.000	6.0000	17.00
Glucose	767.0	121.558018	30.336089	44.000	99.5000	117.000	140.0000	199.00
BloodPressure	767.0	72.389831	12.104228	24.000	64.0000	72.000	80.0000	122.00
SkinThickness	767.0	29.016949	8.426210	7.000	25.0000	29.000	32.0000	63.00
Insulin	767.0	140.692308	86.437570	14.000	121.0000	125.000	127.5000	846.00
BMI	767.0	32.452282	6.879184	18.200	27.5000	32.300	36.6000	67.10
DiabetesPedigreeFunction	767.0	0.471742	0.331524	0.078	0.2435	0.371	0.6265	2.42
Age	767.0	33.203390	11.721879	21.000	24.0000	29.000	41.0000	81.00
Outcome	767.0	0.348110	0.476682	0.000	0.0000	0.000	1.0000	1.00



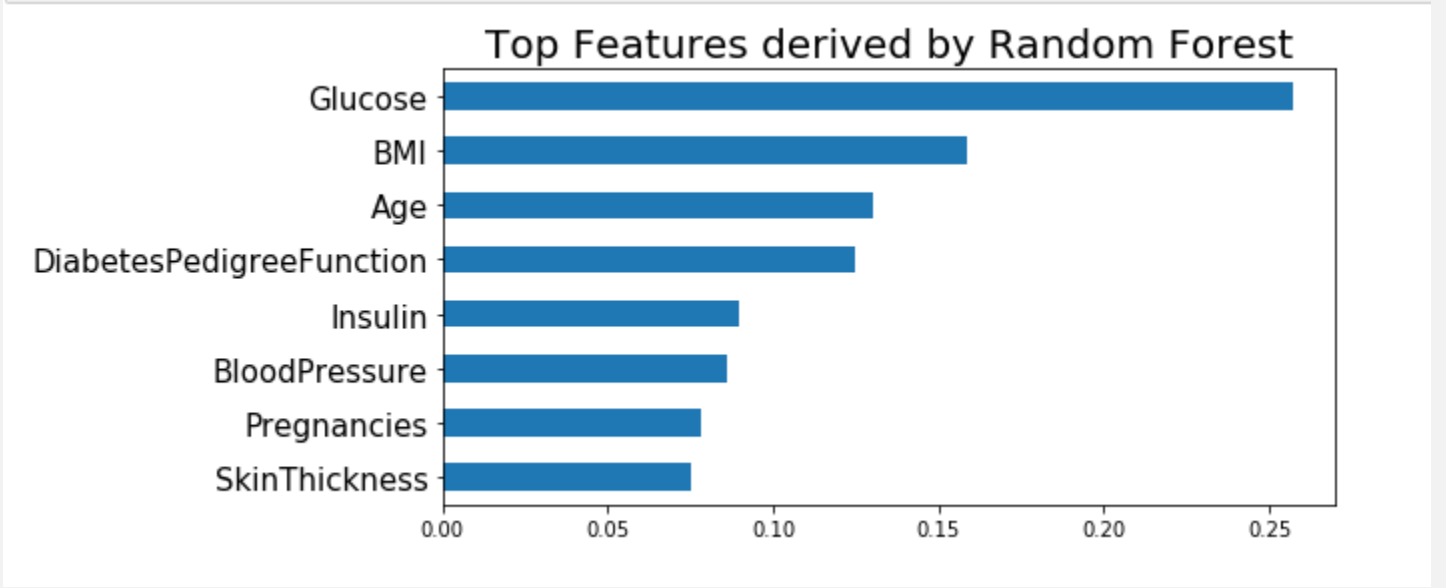
Sagemaker XGBoost doesn't require separate data preprocessing because it is very robust and can handle missing data and class imbalance automatically. As mentioned before, since XGBoost is not

a distance based algorithm, it is not much affected by outliers, whereas distance based algorithm such as Logistic Regression and Support Vector have impact with outliers.

3.2 Implementation

3.2.1 Feature Importance

Performed feature importance study in Prima diabetes dataset. The figure below shows the relative importance of features derived from Random Forest and their contribution to the model. Since it was a small dataset with less columns, I didn't use Feature Selection technique such as PCA we learnt in the course.



From the above feature importance plot we can derive the following:

- Glucose is the most important factor in determining the onset of diabetes followed by BMI and Age.
- Other factors such as Diabetes Pedigree Function, Pregnancies, Blood Pressure, Skin Thickness and Insulin also contributes to the prediction.

As we can see, the results derived from feature importance makes sense as one of the first things that actually is monitored in high-risk patients is the Glucose level. An increased BMI might also indicate a risk of developing Type II Diabetes. Normally, especially in case of Type II Diabetes, there is a high risk of developing as the age of a person increases.

3.2.2 Feature Scaling/Normalization

Features values in the dataset are on very different scales, it is good to scale the feature values to the space between 0-1 for the algorithms to perform better. Currently for all models, implemented the MinMaxScaler and call the fit() function on the training set, then apply the transform() function on the train and test sets to create a normalized version of each dataset to avoid data leakage.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age

	0	1	2	3	4	5	6	7
0	0.058824	1.000000	0.530612	0.642857	0.152055	0.599515	0.560788	0.020408
1	0.117647	0.335484	0.448980	0.107143	0.047945	0.070388	0.241010	0.102041
2	0.529412	0.619355	0.714286	0.392857	0.152055	0.351942	0.278253	0.489796
3	0.000000	0.877419	0.551020	1.000000	0.000000	1.000000	1.000000	0.081633
4	0.000000	0.393548	0.448980	0.267857	0.152055	0.043689	0.065068	0.020408

When I applied *MinMaxScaler/StandardScaler* for train dataset alone for XGBoost, accuracy got dropped by 3% , but it got improved by proper hyperparameter turning. This avoids data leakage as the calculation of the minimum and maximum value for each input variable is calculated using only the training dataset (X_train) instead of the whole dataset (X).

3.2.3 Build the Baseline Model

After pre-processing, Split the dataset by using train_test_split function from the sci-kit learn package to split into train, test and validation data in 80:10:10 ratio to train, test and validate model.

As the dataset have imbalanced class with 500 patients with diabetes negative and 268 with diabetes positive, while training model specified stratify parameter to make a split so that the proportion of values in the splitted dataset produced will have the same proportion as the stratify column provided.

Builded the following six models as baseline to evaluate against model build with AWS Sagmaker XGBoost pre-built container.

1. Bernoulli Naïve Bayes
2. Logistic Regression
3. K Neighbors Classifier
4. Decision Tree Classifier
5. Support Vector Classifier(SVC)
6. Linear SVC

3.2.4 Build the XGBoost Model in Sagemaker container

XGBoost model is builted in AWS Sagemaker by following given below main steps

1. Split data into train/validation/test sets
2. Save splitted data as train.csv,test.csv and validation.csv
3. Load the Training, Test and Validation data to AWS S3 Bucket
4. Create a User IAM role in AWS Sagemaker Instance
5. Create an xgboost Estimator
6. Configure Estimator for training by specifying Algorithm container, instance count, instance type, model output location to train model

7. Create a HyperParameterTuner object
8. Run HyperParameterTuner to search for best performing models by specifying ranges for hyperparameter such as max depth, learning rate, etc
9. Deploy best performing model as SageMaker endpoint

3.3 Refinement

After building an initial XGBoost Model with an experimental initial set of Hyperparameters vlue for *max_depth* as *5* and *learning rate* as *0.2*, able to get result which exceed performance of all baseline model except logistic regression. Then I created and run hyperparameter object by altering the range of values for the following parameters *max_depth* values to range from *3 to 12* and *learning rate* from *0.05 to 0.5* , tune hyperparameter until it can have same result as Logistic regression around 75 % accuracy and ROC score around 83 %

4. Results

4.1 Model Evaluation and Validation

As defined in Metrics section (1.3), I had evaluated Sagemaker XGBoost model with the baseline models based on Accuracy, F1 , ROC-AUC score .

Baseline Models Evalluation Metrics

	model	accuracy	precision	recall	f1score	rocauc	logloss
0	BernoulliNB	0.640077	0.066667	0.004545	0.008511	0.532691	12.111000
1	LogisticRegression	0.758923	0.728651	0.520190	0.604459	0.832284	8.074052
2	KNN	0.765427	0.676133	0.656977	0.665206	0.796619	11.214034
3	DecisionTree	0.696988	0.575270	0.552431	0.557657	0.678512	10.765468
4	SVC	0.749100	0.695265	0.529175	0.599722	0.826883	7.625486
5	LinearSVC	0.758936	0.709570	0.552114	0.618975	0.832705	8.522607

Vs

XGBoost result with initial hyperparameter

	precision	recall	f1-score	support
0	0.79	0.80	0.79	55
1	0.48	0.45	0.47	22
accuracy			0.70	77
macro avg	0.63	0.63	0.63	77
weighted avg	0.70	0.70	0.70	77

ROC_AUC_Score: 0.7880165289256198
Accuracy Score: 0.7012987012987013

XGBoost result by tuning hyperparameter

	precision	recall	f1-score	support
0	0.82	0.84	0.83	55
1	0.57	0.55	0.56	22
accuracy			0.75	77
macro avg	0.70	0.69	0.69	77
weighted avg	0.75	0.75	0.75	77

ROC_AUC_Score: 0.8301652892561983
Accuracy Score: 0.7532467532467533

The final XGBoost model selected after hyperparameter tuning had accuracy value

- Improved by 5 % from initial setting of hyperparameter of XGBoost Model
- Higher than most of the all baseline models BernoulliNB, SVC, Decision tree

- Achieved almost same Accuracy (75%) and ROC-AUC score (83%) as Logistic regression.

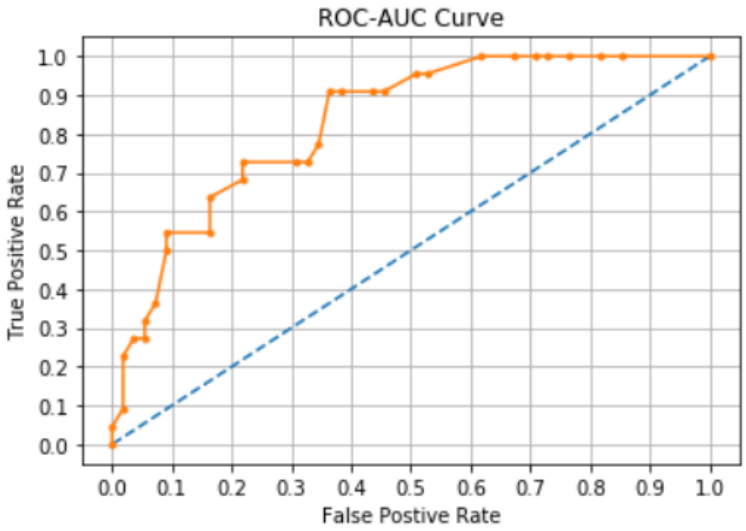
I still consider with tuning learning rate, max depth along with tuning max job value possible to achieve accuracy above 76 % to 80% with XGBoost model in AWS SageMaker.

4.2 Justification

Improved Accuracy, F1 and ROC-AUC score comparing to baseline model, Despite this improvement I would still believe current final XGboost model have the potential to predict with much better accuracy with dataset with more number of observation with having patient of all ages as the current dataset have younger patients with average age of 33 years as per data exploration study(2.1 section) I do think though that the model demonstrates the potential value in collecting a dataset that does not have these limitations and if similar results are achieved, that model being more generally useful as the prediction endpoint from Sagemaker can be easily integrated as API to Diabetes Predictor Web/mobile application .

5. Conclusion

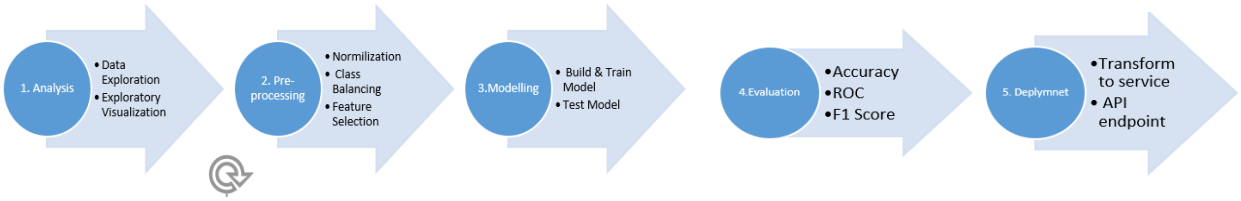
The XGBoost algorithm is ideally suited for classification type of problem. From the given below evaluation ROC-AUC score plot it is clear that XGBoost Model shows that curve had raised steeply covering a large area before reaching the top-right.



Almost 83 % of the test cases is predicted with resulting in high sensitivity and low False positive rate which is very important for any disease predictor

5.1 Reflection

As per basic workflow proposed for the capstone project, completed all the given below process



1. **Analyzed** the problem through visualizations and data exploration to have a better understanding of data and features that are appropriate for solving it by plotting correlation matrix.
2. **Pre-Processed** data, as it plays an important role in improving performance of the model for small dataset such as by replacing null values with mean/median, and selected important features through feature selection.
   As XGBoost is able to perform pre-processing of replacing null values with median, class imbalance and feature selection by itself, implemented pre-processing and feature selection for building base models alone. Splitted the processed dataset into Train and Test dataset (80: 20 ratio) after scaling/normalizing the features with values between 0 to 1
3. **Implemented Models** Baseline Models with scikit library and XGboost in SageMaker by loading Training, Test and Validation data in S3 Bucket, followed by selecting Algorithm Container Registry Path, Configure Estimator for training by specifying Algorithm container, instance count, instance type, model output location to train model.
   Then tuned the model by HyperParameterTuner object by specifying ranges for hyperparameter such as max depth, learning rate, etc to find best performing model from the model search feature of Sagemaker pre-built container of XGBoost.
4. Performed **Evaluation** of model as explained in Evaluation Metric section of this report by computing Accuracy Score, F1 score and ROC-AUC score.
5. Finally **Deployed** model as SageMaker endpoint to run prediction on test data by specifying instance count, instance type and endpoint name in Sagemaker

The XGBoost algorithm uses a technique known as the gradient boosting decision tree algorithm which generates a model, and then combines that with a new model meant to predict the errors of the first model. The model does this iteratively layer upon layer until it reaches a point where no further improvements in performance is made. It works particularly well on structured data and also performs well computationally as it utilizes the parallelism to push the limit of computational resources which leaves low memory footprint and faster training. Also it handles missing values automatically and it has interactive feature analysis feature such as the feature importance analysis which helps in identifying the most important features for the algorithm. It is also robust enough to handle the class imbalance in the dataset. Furthermore, AWS SageMaker has a pre-built container for this algorithm which simplifies the development of an initial model without needing to implement a custom container for a deep learning approach like TensorFlow or PyTorch. Analyzed with hyperparameters of XGBoost with Sagemaker feature to find best performing model. This is very important feature for anyone who may want to reproduce best results without the need for manually searching parameter values to tune model for best performance with some basic knowledge in tuning parameter of model in short time . Thus

XGBoost for diabetes prediction is good choice computationally than Random forest which proven to give high accuracy for pima diabetes dataset observed from many Kaggle diabetes prediction project .

5.2 Improvement

Able to achieve good result through XGBoost model implemented with AWS Sagemaker pre-built container which is better than multiple baseline models with pre-processed dataset and I still believe, it is possible to improve further by varying parameter range of various parameters of XGBoost through hyperparameter tuning.

One other consideration for this dataset is that it is collected specifically from a study using members of the Pima Indian tribe. It is entirely conceivable that there are genetic or lifestyle differences within this population that make any model built using this data not robust enough or valid for a larger more general population. With this in mind, any resulting model would need to be validated against populations of patients of either their respective population subgroup or against the general population before being widely deployed or used. In future I also would like to try this model on the dataset of diabetes with 2000 patient's data, taken from the hospital Frankfurt (https://www.kaggle.com/johndasilva/diabetes) also to compare the performance of model with benchmark Prima Indian Diabetes Dataset which also has same proportion of true cases (684 – 34.20% and false cases (1316 – 65.80%) . Both the dataset consists of same eight medical distinct measurement variables.

Possible to enhance model as End to End web/mobile solution with Flask integrating via API to AWS SageMaker Predictor endpoint

References

1. Wu Z, McGoogan JM. Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China: summary of a report of 72 314 cases from the Chinese Center for Disease Control and Prevention. JAMA. 2020. https://doi.org/10.1001/jama.2020.2648.
2. The Pima Indian Diabetes Dataset used originally came from this paper: ** Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261--265). IEEE Computer Society Press. Now available for download via Kaggle here.
3. N.Sneha ,Tarun Gangil Analysis of diabetes mellitus for early prediction using optimal features selection
4. Deepti Sisodia , Dilip Singh Sisodia. Prediction of Diabetes using Classification Algorithms International Conference on Computational Intelligence and Data Science (ICCIDS 2018)
5. https://www.kaggle.com/vipulgandhi/how-to-choose-right-metric-for-evaluating-ml-model
6. http://www.kmdatascience.com/2017/07/k-folds-cross-validation-in-python.html