NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA

LA-2 REPORT

on

Healthcare AI model using XAI techniques

Submitted in partial fulfilment of the requirement for the award of Degree of

Bachelor of Engineering

in

Information Science and Engineering

Submitted by:

L Shriya Reddy **1NT21IS082**

Under the Guidance of

Dr. K Aditya Shastry

Professor, Dept. of ISE, NMIT

NITTE EDUCATION TRUST

Department of Information Science and Engineering

(Accredited by NBA Tier-1)

2024-2025

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM)

Table of Contents

PROBLEM STATEMENT

Build a Python application to interpret the predictions of a healthcare AI model using XAI techniques and evaluate its effectiveness in clinical decision-making.

INTRODUCTION

In the realm of healthcare, AI models are increasingly being used to support clinical decision-making by providing predictive insights based on patient data. This project aims to build a Python application that interprets the predictions of a healthcare AI model using Explainable AI (XAI) techniques, specifically focusing on predicting diabetes. The effectiveness of these models is evaluated in terms of their interpretability and accuracy, crucial factors in clinical settings where understanding the rationale behind predictions can significantly impact patient care.

For this project, we employ two robust machine learning algorithms: Random Forest and Gradient Boosting. Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It provides high accuracy, handles large datasets with higher dimensionality, and is less prone to overfitting compared to a single decision tree. On the other hand, Gradient Boosting is another powerful ensemble technique that builds models sequentially, with each new model attempting to correct errors made by the previous ones. It combines the predictions from several weaker models to produce a stronger model, often leading to improved predictive performance.

LIME (Local Interpretable Model-agnostic Explanations) is employed to make the predictions of these models interpretable. LIME explains the predictions of any classifier by learning an interpretable model locally around the prediction. This approach allows clinicians to understand which features contribute most to the prediction of diabetes for a particular patient, thus enhancing trust and transparency in AI-driven decision support systems.

ALGORITHM USED:

Random Forest Algorithm

1. Bootstrap Sampling: Randomly select \(n\) samples with replacement from the training data to create \(m\) different subsets (bootstrap samples).

2. Decision Trees: For each bootstrap sample, grow an unpruned decision tree by:

- Randomly selecting \(k\) features from the total \(p\) features at each split.
- Splitting the node using the best feature among the \(k\) selected features.

3. Voting/Averaging:

- For classification: Each tree votes for a class, and the majority vote is the final prediction.
- For regression: The predictions from all trees are averaged to give the final prediction.

Gradient Boosting Algorithm

1. Initialize Model: Start with a base model (e.g., a simple decision tree) that predicts the mean of the target variable.

2. Compute Residuals: Calculate the residuals (errors) between the actual values and the model's predictions.

3. Fit Weak Learner: Train a new weak learner (e.g., a shallow decision tree) to predict the residuals.

4. Update Model: Add the predictions of the new weak learner to the existing model's predictions, scaled by a learning rate \(\alpha\).

5. Iterate: Repeat steps 2-4 for a specified number of iterations or until convergence.

METHODOLOGY

The project follows a structured methodology to build, evaluate, and interpret the AI models:

1. **Data Preparation**: The dataset used is the diabetes dataset, which is loaded into a pandas Data Frame. The data is checked for null values, and rows with zero values in critical features like Blood Pressure, BMI, and Glucose are removed to ensure data quality.
2. **Feature Selection and Splitting**: The features relevant to diabetes prediction, such as Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and Age, are selected. The dataset is then split into training and testing sets using stratified sampling to maintain the proportion of diabetic and non-diabetic cases.
3. **Model Training and Evaluation**: Two models, Random Forest and Gradient Boosting, are trained on the training data. Their performance is evaluated on the test data using accuracy as the metric. The evaluation results are compared to select the better-performing model.
4. **Explainable AI with LIME**: LIME is used to interpret the predictions of the trained models. A LIME explainer is created, and for a selected instance, LIME provides a local explanation, highlighting the contribution of each feature to the prediction.

CODE

Import Libraries:

[11]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import lime
```

[13]:
```
!pip install lime
```

[15]:
```
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import lime
```

[16]:
```
#import seaborn as sns
#import lime.lime_tabular
from lime import lime_tabular
from lime import submodular_pick
```

[17]:
```
df =pd.read_csv('diabetes.csv' )
df.columns
```

[17]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[18]:
```
df.head()
```

[18]:

	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

[19]:
```
print("Diabetes Data Set Dimensions: {} ".format (df.shape))
```

```
Diabetes Data Set Dimensions: (768, 9)
```

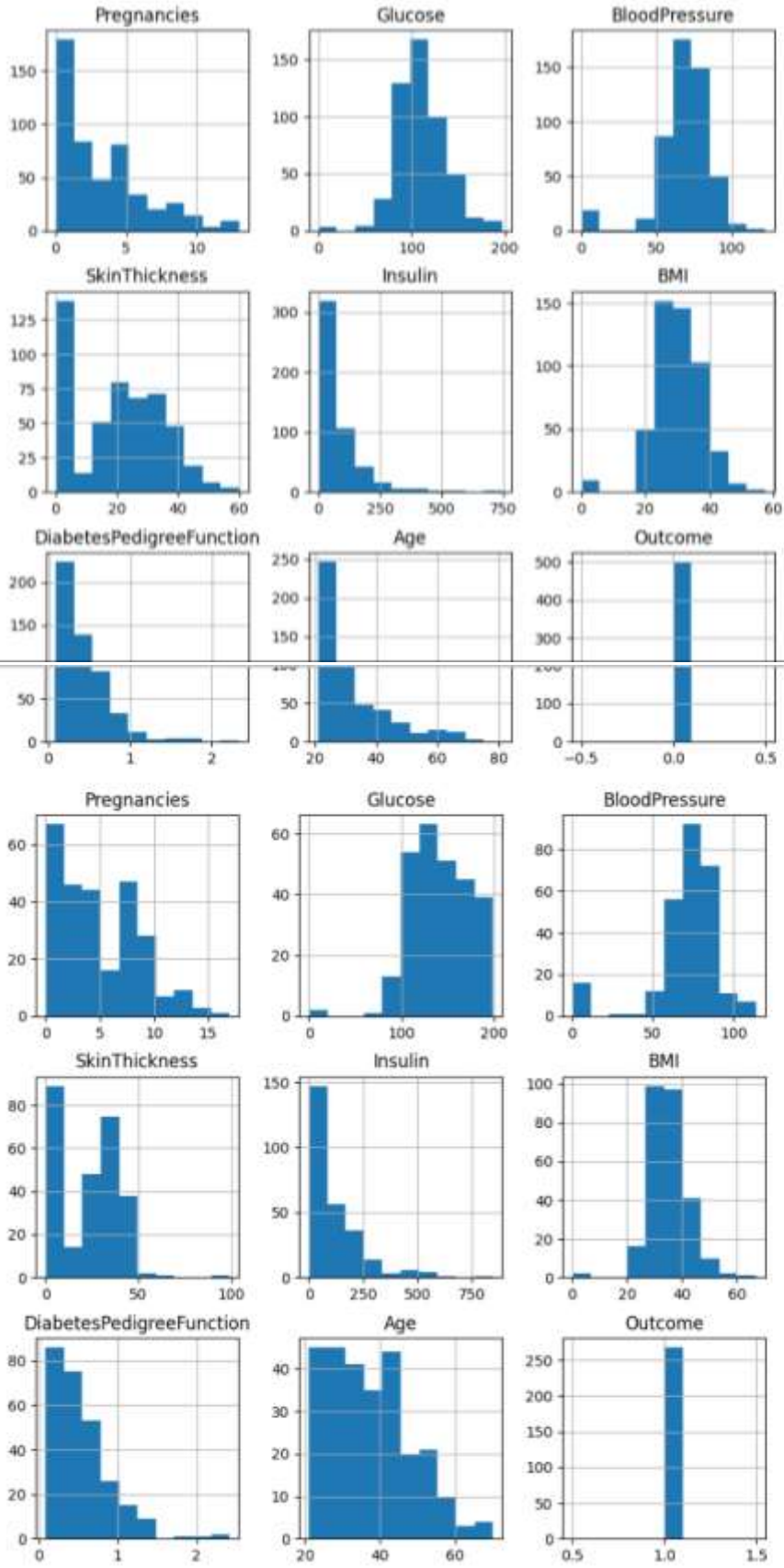
[20]:
```
df.groupby('Outcome' ) .size()

df.groupby('Outcome'). hist( figsize=(9, 9))

df.isnull().sum()

df_mod = df[(df.BloodPressure != 0) & (df.BMI != 0) &(df.Glucose != 0)]
print(df_mod.shape)
```

```
(724, 9)
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome

[21]:
```
feature_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']
x = df_mod[feature_names]
y = df_mod.Outcome
```

[22]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
models = []
models. append(('RF',RandomForestClassifier ( )))
models. append(( 'GB', GradientBoostingClassifier( )))
#Evaluation Methods
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

[23]:
```
X_train, X_test, y_train, y_test = train_test_split(x, y, stratify = df_mod.Outcome, random_state=0)
names = [ ]
scores  = [ ]
for name, model in models:
 model.fit(X_train, y_train)
 y_pred = model.predict(X_test)
 scores.append(accuracy_score(y_test, y_pred))
 names.append(name)
tr_split = pd.DataFrame({'Name' : names, 'Score': scores})
print(tr_split)
```

```
  Name     Score
0   RF  0.762431
1   GB  0.773481
```

[24]:
```
np.random.seed(123)
```

[25]:
```
predict_fn = lambda x: model.predict_proba(x)
```

[26]:
```
X = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]
X_featurenames = X.columns
```

[27]:
```
myexplainer = lime.lime_tabular.LimeTabularExplainer(np.array(X_train),
feature_names = X_featurenames,
class_names=['No Diabetes', 'Diabetes'],
```

```
myexplainer = lime.lime_tabular.LimeTabularExplainer(np.array(X_train),
feature_names = X_featurenames,
class_names=['No Diabetes', 'Diabetes'],
categorical_features='Outcome',
verbose=True, mode='classification')
```

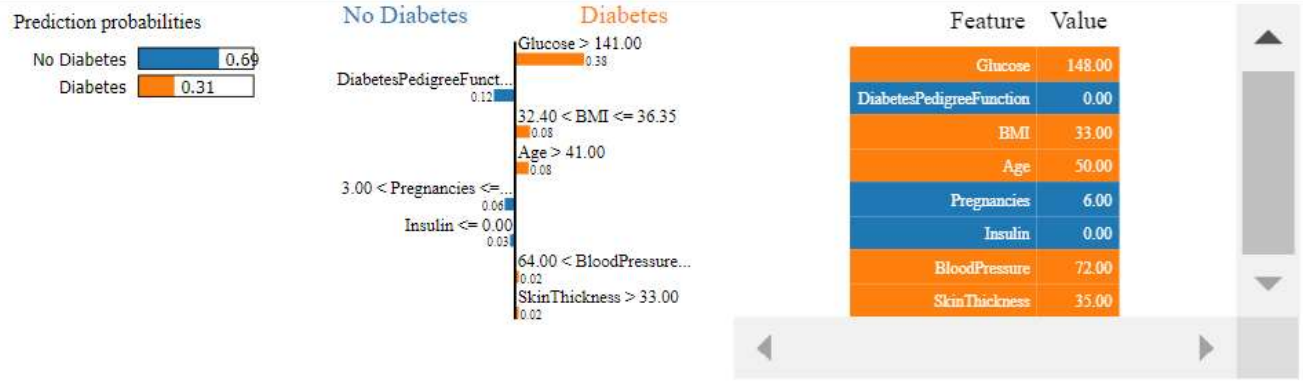
```
i=0
exp = myexplainer.explain_instance(df.loc[i,feature_names].astype(int).values, predict_fn, num_features=8)
```

```
Intercept 0.2892730837551675
Prediction_local [0.66923176]
Right: 0.30782926432416
C:\Python311\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but GradientBoostingClassifier was fitted with
feature names
  warnings.warn(
```

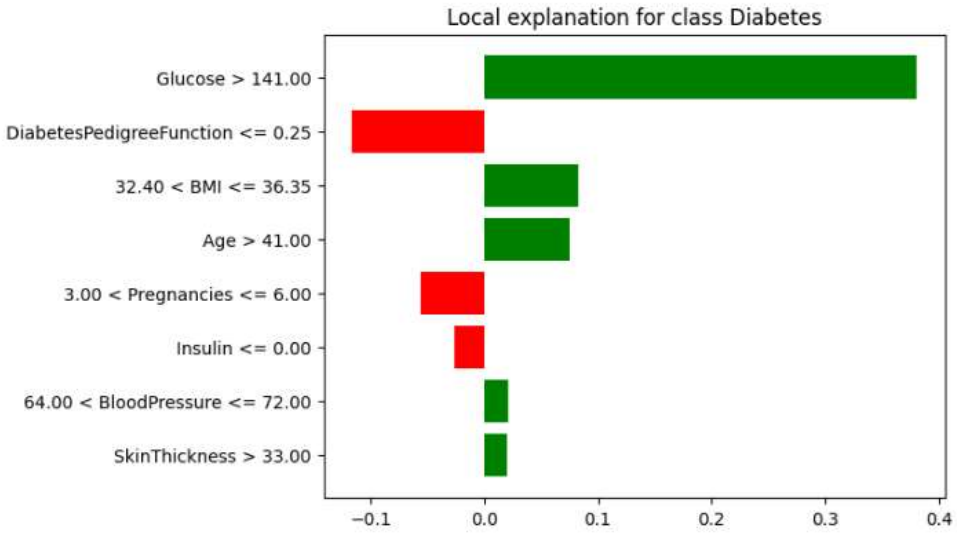
```
exp.show_in_notebook(show_table=True)
```

OUTPUT

```
[29]: exp.show_in_notebook(show_table=True)
```




```
[30]: exp.as_list()
figure = exp.as_pyplot_figure(label = exp.available_labels()[0])
```



CONCLUSION

The project successfully demonstrates the application of Random Forest and Gradient Boosting algorithms to predict diabetes and uses LIME to provide interpretability to these predictions. The visualizations generated by LIME show the contribution of each feature towards the model's prediction, making it easier for healthcare professionals to understand and trust the AI system's decisions. The histograms grouped by 'Outcome' visually differentiate the distributions of features for diabetic and non-diabetic patients, providing insights into the data. This interpretability is crucial for effective clinical decision-making, ensuring that AI-driven predictions are transparent and actionable.