GlucoSense- AI-Powered Diabetes Detection for Early Intervention

Title: Identification of Diabetes in a person based on healthcare statistics.

- Project Statement:

  Diabetes cases over the past fifteen years have bloomed all over the world. Lifestyle plays a very important role in it. In recent years, there has been an improvement in awareness regarding the health effects of diabetes. This has led to people getting themselves tested for diabetes than they would have earlier, as its risk can be reduced if it is predicted early.

- Outcomes:

  The goal is to develop a model which helps better understand the relationship between lifestyle and diabetes and help predict whether a patient has diabetes, is pre-diabetic or healthy.

➢ **Objective**

The main objectives of the project are:

- Data Collection and Preprocessing: Gather and clean the dataset to ensure it's suitable for model training.

- Model Development: Apply various machine learning algorithms to predict diabetes status.

- Model Evaluation: Assess the performance of the models using evaluation metrics such as accuracy, precision, recall, and F1-score.

- Feature Importance: Identify the most significant factors contributing to diabetes risk.

Modules to be implemented

1. Data Collection – Statistics of healthcare and lifestyle survey information
2. Data Exploration (EDA) and Data Preprocessing
3. Feature Selection and dimension reduction approaches
4. Build a classification model
5. Evaluation metrics
6. Presentation and Documentation

1. Data collection:

We needed dataset including various features with respect to diabetic symptoms. So we took a dataset from Kaggle.com having 16 different features.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	age	gender	polyuria	polydipsia	sudden_we	weakness	polyphagia	genital_thr	visual_blurr	itching	irritability	delayed_he	partial_pare	muscle_stif	alopecia	obesity	class
2	40	Male	0	1	0	1	0	0	0	1	0	1	0	1	1	1	1
3	58	Male	0	0	0	1	0	0	1	0	0	0	1	0	1	0	1
4	41	Male	1	0	0	1	1	0	0	1	0	1	0	1	1	0	1
5	45	Male	0	0	1	1	1	1	0	1	0	1	0	0	0	0	1
6	60	Male	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
7	55	Male	1	1	0	1	1	0	1	1	0	1	0	1	1	1	1
8	57	Male	1	1	0	1	1	1	0	0	0	1	1	0	0	0	1
9	66	Male	1	1	1	1	0	0	1	1	1	0	1	1	0	0	1
10	67	Male	1	1	0	1	1	1	0	1	1	0	1	1	0	1	1
11	70	Male	0	1	1	1	1	0	1	1	1	0	0	0	1	0	1
12	44	Male	1	1	0	1	0	1	0	0	1	1	0	1	1	0	1
13	38	Male	1	1	0	0	1	1	0	1	0	1	0	1	0	0	1
14	35	Male	1	0	0	0	1	1	0	0	1	1	0	0	1	0	1
15	61	Male	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1
16	60	Male	1	1	0	1	1	0	1	1	0	1	1	0	0	0	1
17	58	Male	1	1	0	1	1	0	0	0	0	1	1	1	0	0	1
18	54	Male	1	1	1	1	0	1	0	0	0	1	0	1	0	0	1
19	67	Male	0	1	0	1	1	0	1	0	1	1	1	1	1	1	1
20	66	Male	1	1	0	1	1	0	1	0	0	0	1	1	0	0	1
21	43	Male	1	1	1	1	0	1	0	0	0	0	0	0	0	0	1
22	62	Male	1	1	0	1	1	0	1	0	1	0	1	1	0	0	1
23	54	Male	1	1	1	1	1	1	1	1	0	1	0	1	1	0	1
24	39	Male	1	0	1	0	0	1	0	1	1	0	0	0	1	0	1
25	48	Male	0	1	1	1	0	0	1	1	1	1	0	0	0	0	1

Libraries used:

1. Importing Libraries

```
[21]  1  import matplotlib.pyplot as plt
      2  import numpy as np
      3  import pandas as pd
      4  import seaborn as sns
      5  from sklearn.model_selection import train_test_split, GridSearchCV
      6  from sklearn.ensemble import RandomForestClassifier, ExtraTreesClassifier, GradientBoostingClassifier
      7  from sklearn.svm import SVC
      8  from sklearn.tree import DecisionTreeClassifier
      9  from sklearn.linear_model import LogisticRegression, Lasso
     10  from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, mean_squared_error, r2_score
     11  from sklearn.feature_selection import SelectKBest, chi2
     12  from sklearn.preprocessing import MinMaxScaler, StandardScaler, LabelEncoder
     13  from sklearn.decomposition import PCA
     14  from sklearn.neighbors import LocalOutlierFactor
     15  import xgboost as xgb

[22]  1  import warnings
      2  warnings.filterwarnings("ignore", category=UserWarning)
```

Local Outlier Factor

```
Outlier indices detected:
Index([  2,  10,  12,  19,  25,  31,  68,  96,  97, 100, 101, 102, 103, 108,
       111, 113, 116, 119, 128, 131, 132, 133, 147, 151, 167, 181, 184, 185,
       186, 187, 193, 195, 217, 219, 235, 238, 275, 277, 284, 286, 372, 374,
       385, 424, 430, 436, 463, 465, 472, 474, 518],
      dtype='int64')
```

Encoding for Gender

	gender
0	0
1	0
2	0
3	0
4	0

dtype: int64

2.Data Exploration (EDA) and Data Preprocessing

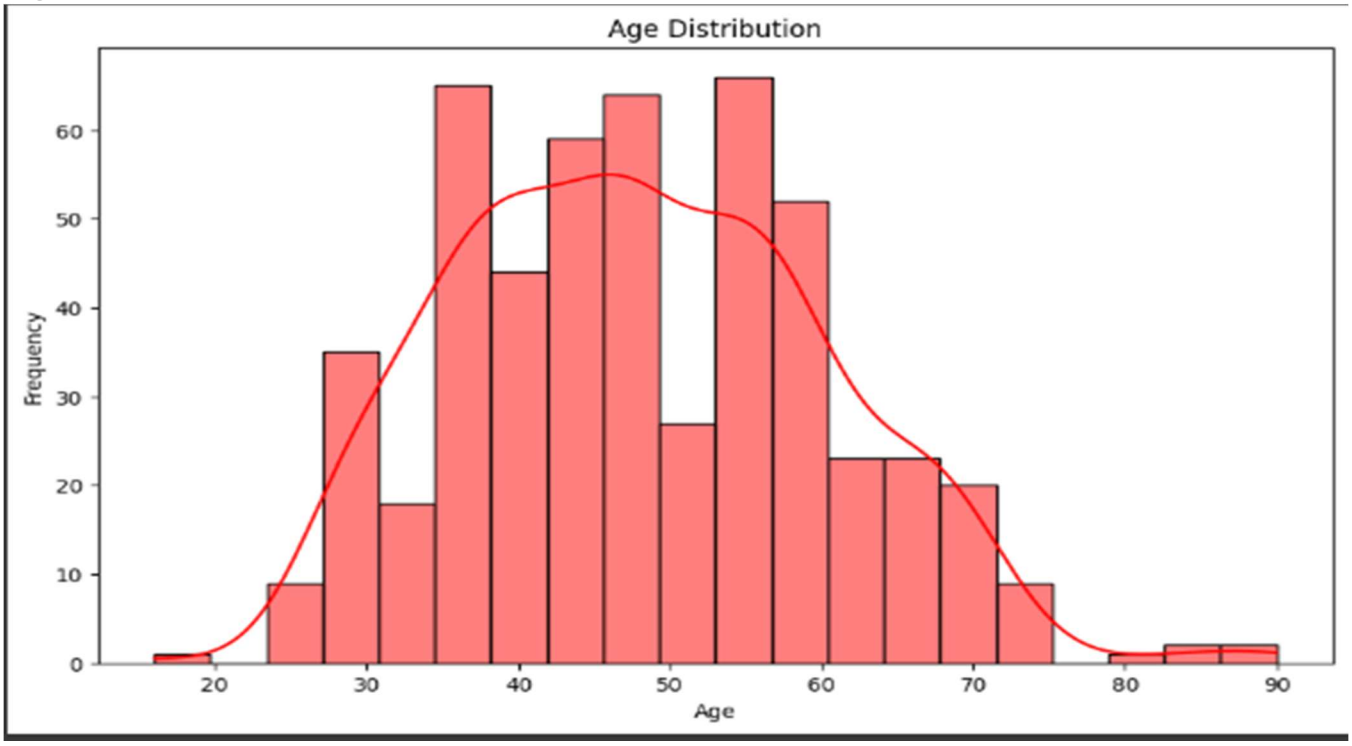
Checking for Missing values:

```
df.isnull().sum()
```

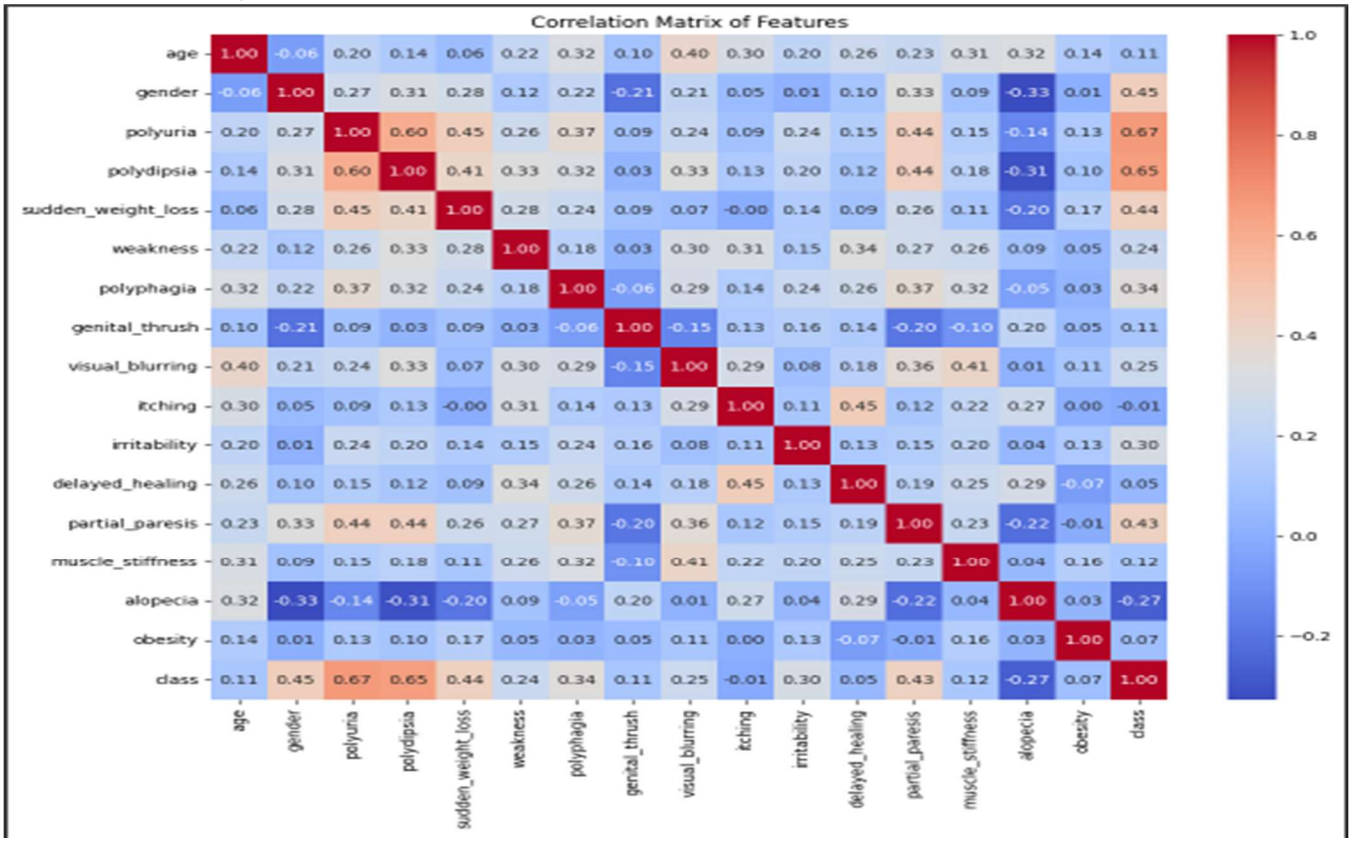
	0
age	0
gender	0
polyuria	0
polydipsia	0
sudden_weight_loss	0
weakness	0
polyphagia	0
genital_thrush	0
visual_blurring	0
itching	0
irritability	0
delayed_healing	0
partial_paresis	0
muscle_stiffness	0
alopecia	0
obesity	0
class	0

dtype: int64

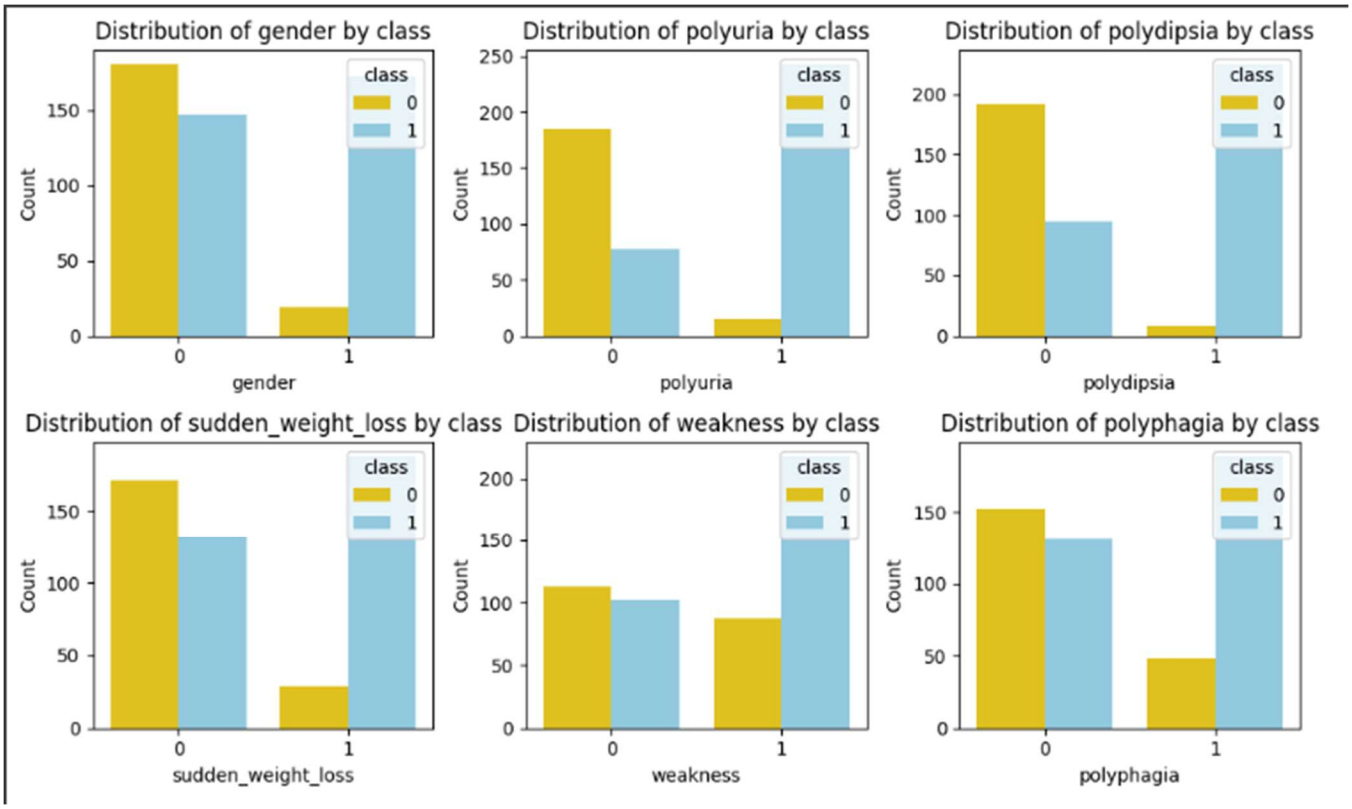
Age Distribution:



Bivariate Analysis:



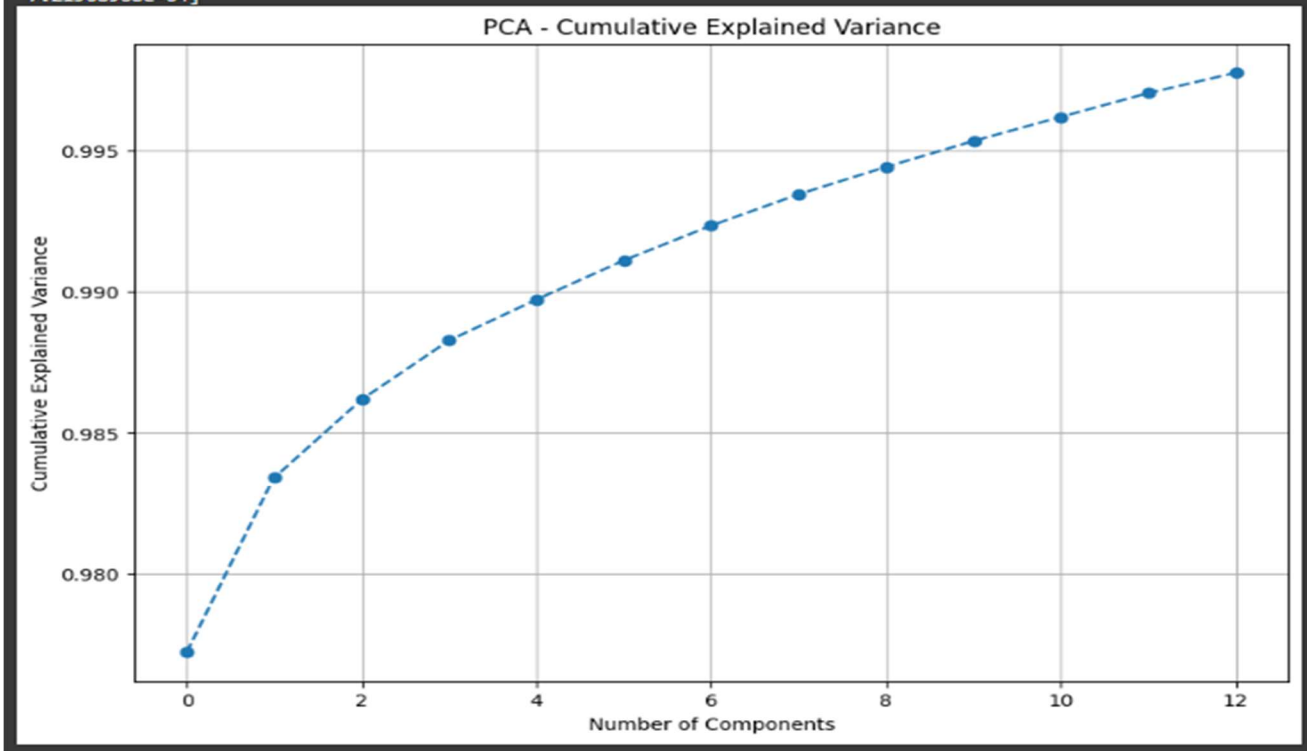
Distribution of features Based on Positive Diabetes patients



The output is a series of visualizations that show how each categorical and numerical variable is distributed with respect to the diabetes status (or whatever the target column represents). This helps in understanding how different features relate to the target variable, which can be valuable for further analysis and model building.
Categorical Features: Count plots showing the distribution of each categorical feature split by the target class, with bars colored according to the class values.
Numerical Features: Box plots showing the distribution of each numerical feature split by the target class, helping to visualize the spread and central tendencies.

5. Dimentionality Reduction

```
Explained variance ratio: [9.77204729e-01 6.20344006e-03 2.77177453e-03 2.09466111e-03
 1.44905008e-03 1.38980290e-03 1.23388624e-03 1.11028460e-03
 9.69208046e-04 9.18306406e-04 8.55337312e-04 8.43022421e-04
 7.21968968e-04]
```



6. Feature Selection

Chi square:

```
        age  gender  polyuria  polydipsia  sudden_weight_loss  weakness  \
0  0.324324     0.0       0.0         1.0                 0.0       1.0
1  0.567568     0.0       0.0         0.0                 0.0       1.0
2  0.337838     0.0       1.0         0.0                 0.0       1.0
3  0.391892     0.0       0.0         0.0                 1.0       1.0
4  0.594595     0.0       1.0         1.0                 1.0       1.0

   polyphagia  genital_thrush  visual_blurring  itching  irritability  \
0         0.0             0.0              0.0      1.0           0.0
1         0.0             0.0              1.0      0.0           0.0
2         1.0             0.0              0.0      1.0           0.0
3         1.0             1.0              0.0      1.0           0.0
4         1.0             0.0              1.0      1.0           1.0

   delayed_healing  partial_paresis  muscle_stiffness  alopecia  obesity
0              1.0              0.0               1.0       1.0      1.0
1              0.0              1.0               0.0       1.0      0.0
2              1.0              0.0               1.0       1.0      0.0
3              1.0              0.0               0.0       0.0      0.0
4              1.0              1.0               1.0       1.0      1.0
Selected feature indices: [ 1  2  3  4 12]
Selected feature names: Index(['gender', 'polyuria', 'polydipsia', 'sudden_weight_loss',
       'partial_paresis'],
      dtype='object')
Transformed feature matrix shape: (520, 5)
```

Regularization:

Using LASSO:

```
Mean Squared Error: 0.15
R-squared: 0.33
Feature1     0.000714
Feature2     0.000000
Feature3     0.187976
Feature4     0.113106
Feature5     0.000000
Feature6     0.000000
Feature7     0.000000
Feature8     0.000000
Feature9     0.000000
Feature10   -0.000000
Feature11    0.000000
Feature12    0.000000
Feature13    0.000000
Feature14    0.000000
Feature15   -0.000000
Feature16    0.000000
dtype: float64
lasso score: 0.3268598571924092
```

Modelling:

The models used are as follows

1. Decision Tree (DT)
2. Gradient Boosting (GB)
3. Logistic Regression (LR)
4. Random Forest (RF)
5. Support Vector Machine (SVM)
6. Extra Trees (ET)
7. XGBoost (XGB)

Training_Test_Split:

Train_Test_Split

```
# Here, we split the data into 70% training and 30% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Display the shapes of the resulting datasets
print(f"Training features shape: {X_train.shape}")
print(f"Testing features shape: {X_test.shape}")
print(f"Training target shape: {y_train.shape}")
print(f"Testing target shape: {y_test.shape}")
```

```
Training features shape: (364, 16)
Testing features shape: (156, 16)
Training target shape: (364,)
Testing target shape: (156,)
```

Models (Without hyper paramete tuning)

	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.842105	0.849492	0.842105	0.834334
1	Support Vector Machine	0.644737	0.415686	0.644737	0.505474
2	Decision Tree	0.881579	0.881199	0.881579	0.879855
3	Extra Trees	0.921053	0.923730	0.921053	0.919443
4	Random Forest	0.947368	0.947864	0.947368	0.946883
5	Gradient Boosting	0.894737	0.896118	0.894737	0.892590
6	XGBoost	0.855263	0.856038	0.855263	0.851392

Highest Accuracy and Performance:
Random Forest is the top performer with the highest accuracy (0.947), precision (0.948), recall (0.947), and F1-Score (0.947), indicating a highly reliable and effective model.
Strong Performers:

Extra Trees and Gradient Boosting also show strong performance with high metrics across the board. Extra Trees have an accuracy of 0.921 and Gradient Boosting has an accuracy of 0.895.
Decision Tree demonstrates good performance as well, with metrics close to those of Extra Trees and Gradient Boosting.
Moderate Performance:
XGBoost has solid performance, but slightly lower than the top models, with an accuracy of 0.855.
Logistic Regression performs reasonably well with an accuracy of 0.842.
Lowest Performance:
Support Vector Machine (SVM) shows the lowest performance among the models, with an accuracy of 0.645, indicating it may not be the best choice for this dataset.

Applying Hyper parameter Tuning on the models

1.Logistic regression:

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
Best parameters found: {'C': 1, 'max_iter': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
Best cross-validation score: 0.8742857142857142
Accuracy: 0.84
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.63      0.74        27
           2       0.82      0.96      0.89        49

    accuracy                           0.84        76
   macro avg       0.86      0.79      0.81        76
weighted avg       0.85      0.84      0.83        76
```

The model performed quite well, achieving an accuracy of 84% on the test set. The precision and recall metrics indicate that the model is quite reliable in predicting both classes, especially class 2, where recall is exceptionally high (96%). The classification report provides a balanced view of the model's performance across different metrics, showing that it effectively balances precision and recall.

2.Decision Tree

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best parameters found: {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'splitter': 'random'}
Best cross-validation score: 0.8800000000000001
Accuracy: 0.88
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        27
           2       0.92      0.90      0.91        49

    accuracy                           0.88        76
   macro avg       0.87      0.87      0.87        76
weighted avg       0.88      0.88      0.88        76
```

The model achieved an accuracy of 88% on the test set.
Class 2 predictions are particularly strong with high precision (0.92) and recall (0.90).
Class 0 shows a balanced performance with good precision (0.82) and recall (0.85).

3.Random Forest

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best parameters found: {'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 50}
Best cross-validation score: 0.9257142857142858
Accuracy: 0.95
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.89      0.92        27
           2       0.94      0.98      0.96        49

    accuracy                           0.95        76
   macro avg       0.95      0.93      0.94        76
weighted avg       0.95      0.95      0.95        76
```

The model achieved an accuracy of 95% on the test set, indicating strong performance.
Class 2 predictions are highly accurate, with a precision of 0.94 and recall of 0.98.
Class 0 also shows good performance, with a precision of 0.96 and recall of 0.89.
The high values in precision, recall, and F1-score across both classes suggest that the model generalizes well and is effective in distinguishing between the classes.

4.Gradient Boosting

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
Best parameters found: {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}
Best cross-validation score: 0.9142857142857144
Accuracy: 0.88
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.85      0.84        27
           2       0.92      0.90      0.91        49

    accuracy                           0.88        76
   macro avg       0.87      0.87      0.87        76
weighted avg       0.88      0.88      0.88        76
```

The model achieved an accuracy of 88% on the test set, demonstrating solid performance.

For Class 0, the model has a precision of 0.82, recall of 0.85, and F1-score of 0.84, indicating balanced detection.
For Class 2, the precision is 0.92, recall is 0.90, and F1-score is 0.91, showing strong performance in identifying instances of this class.
The macro and weighted averages provide a good overall performance summary, with values close to each other, suggesting consistent model performance across classes.

5.SVC

```
Fitting 5 folds for each of 972 candidates, totalling 4860 fits
Best parameters found: {'learning_rate': 0.1, 'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 200, 'subsample': 0.8}
Best cross-validation score: 0.9257142857142858
Accuracy: 0.89
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.81      0.85        27
           2       0.90      0.94      0.92        49

    accuracy                           0.89        76
   macro avg       0.89      0.88      0.88        76
weighted avg       0.89      0.89      0.89        76
```

The model achieved an accuracy of 89% on the test set, indicating strong performance.
For Class 0, the model has a precision of 0.88, recall of 0.81, and F1-score of 0.85, reflecting balanced detection with slightly lower recall.
For Class 2, the model shows high performance with a precision of 0.90, recall of 0.94, and F1-score of 0.92.
The macro and weighted averages indicate consistent performance across both classes, with all metrics close to each other.

6.Extra Trees

```
Fitting 5 folds for each of 216 candidates, totalling 1080 fits
Best parameters found: {'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best cross-validation score: 0.9371428571428572
Accuracy: 0.92
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.81      0.88        27
           2       0.91      0.98      0.94        49

    accuracy                           0.92        76
   macro avg       0.93      0.90      0.91        76
weighted avg       0.92      0.92      0.92        76
```

The model achieved an accuracy of 92% on the test set, indicating very strong performance.
For Class 0, the model has a high precision of 0.96, recall of 0.81, and F1-score of 0.88, reflecting excellent identification with slightly lower recall.
For Class 2, the model shows excellent performance with a precision of 0.91, recall of 0.98, and F1-score of 0.94.

The macro and weighted averages suggest balanced and consistent performance across both classes.

7.XGboost

```
Fitting 5 folds for each of 972 candidates, totalling 4860 fits
Best parameters found: {'colsample_bytree': 0.8, 'learning_rate': 0.01, 'max_depth': 3, 'min_child_weight': 1, 'n_estimators': 200, 'subsample': 0.9}
Best cross-validation score: 0.9085714285714286
Accuracy: 0.91
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.74      0.85        27
           1       0.88      1.00      0.93        49

    accuracy                           0.91        76
   macro avg       0.94      0.87      0.89        76
weighted avg       0.92      0.91      0.90        76
```

The model achieved an accuracy of 86% on the test set, indicating good overall performance.
For Class 0, the precision is high at 0.90, but the recall is relatively lower at 0.67, suggesting that while most identified positives are true, many true positives are missed.
For Class 1, the model performs very well with a precision of 0.84, recall of 0.96, and F1-score of 0.90, indicating strong identification of this class.
The macro and weighted averages show balanced performance, though the recall for Class 0 indicates room for improvement.

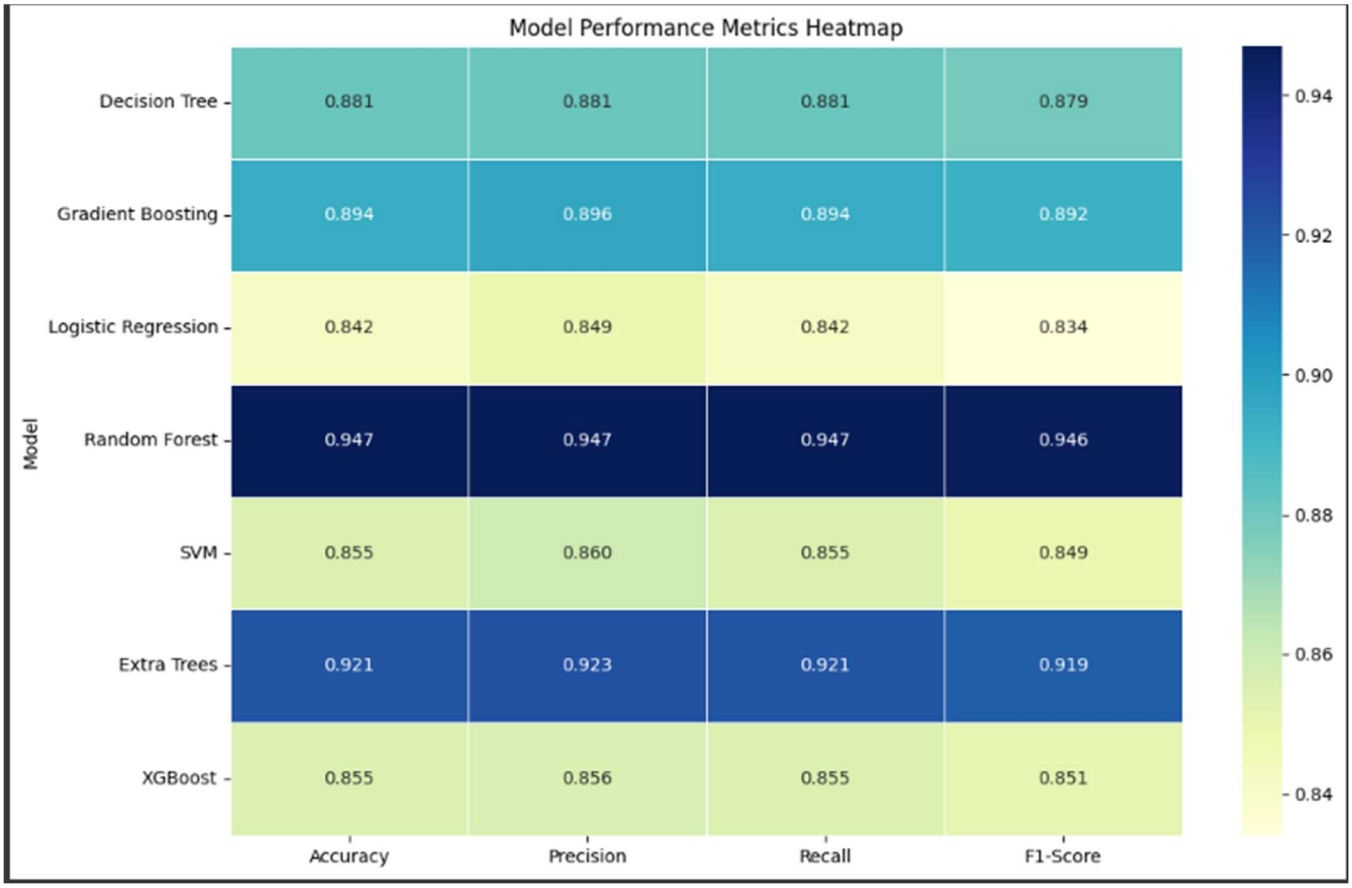
Comparing All the models

	Model	Accuracy	Precision	Recall	F1-Score
0	Decision Tree	0.881579	0.881199	0.881579	0.879855
1	Gradient Boosting	0.894737	0.896118	0.894737	0.892590
2	Logistic Regression	0.842105	0.849492	0.842105	0.834334
3	Random Forest	0.947368	0.947864	0.947368	0.946883
4	SVM	0.855263	0.860855	0.855263	0.849309
5	Extra Trees	0.921053	0.923730	0.921053	0.919443
6	XGBoost	0.855263	0.856038	0.855263	0.851392

	Model	Accuracy	Precision	Recall	F1-Score
0	Decision Tree	88.1	88.1	88.1	87.9
1	Gradient Boosting	89.4	89.6	89.4	89.2
2	Logistic Regression	84.2	84.9	84.2	83.4
3	Random Forest	94.7	94.7	94.7	94.6
4	SVM	85.5	86	85.5	84.9
5	Extra Trees	92.1	92.3	92.1	91.9
6	XGBoost	85.5	85.6	85.5	85.1

Accuracy:
This metric represents the proportion of true results (both true positives and true negatives) among the total number of cases examined.
Highest Accuracy: Random Forest (0.947)
Lowest Accuracy: Logistic Regression (0.842)
Precision:
Precision is the ratio of correctly predicted positive observations to the total predicted positives. It indicates how many of the predicted positive cases were actually true positives.
Highest Precision: Random Forest (0.948)
Lowest Precision: Logistic Regression (0.849)
Recall:
Recall (or Sensitivity) is the ratio of correctly predicted positive observations to all observations in the actual class. It shows how well the model captures all the positive instances.
Highest Recall: Random Forest (0.947)
Lowest Recall: Logistic Regression (0.842)
F1-Score:
The F1-Score is the harmonic mean of Precision and Recall. It provides a single measure of a model's accuracy and is useful when the dataset has an uneven class distribution.
Highest F1-Score: Random Forest (0.947)
Lowest F1-Score: Logistic Regression (0.834)

Performance matrix Heatmap



Highest Accuracy and Performance: Random Forest is the top performer, achieving the highest accuracy (94.7), precision (94.7), recall (94.7), and F1-Score (94.6). This indicates it is the most reliable and effective model for this dataset.

Strong Performers: Extra Trees and Gradient Boosting also exhibit strong performance, with Extra Trees having an accuracy of 92.1 and Gradient Boosting an accuracy of 89.4. Both models maintain high values in precision, recall, and F1-score.
Decision Tree shows a good performance with consistent values close to the top models.

Moderate Performance: XGBoost and SVM deliver solid performance, with accuracies of 85.5. XGBoost is slightly stronger in precision (85.6) compared to SVM's precision (86.0).

Lowest Performance: Logistic Regression shows the lowest performance in this comparison, with an accuracy of 84.2, although it still provides reasonably good metrics.
Conclusion

Random Forest stands out as the most effective model, demonstrating the highest overall metrics.
Among all the metrics we will be taking F1-score into consideration as it is a kind of best fit for the unbalanced dataset.