Early Diabetes Risk Classification

MUHAMMAD RANDA YANDIKA

Outline

- USECASE SUMMARY
- DATA UNDERSTANDING
- EDA & DATA VISUALIZATION
- DATA PREPROCESSING
- MODELLING DATA & MODEL EVALUATION
- CONCLUSION

Use Case Summary

Use case summary

OBJECTIVE

- What factor will cause early diabetes?
- What machine learning algorithms are suitable for predicting diabetes?
- Create models to predict diabetes risk with machine learning techniques.
- Determining the most important factor on the model created for predicting diabetes

OUTCOME

- Identification of the factor that causes early diabetes.
- Machine learning algorithms that are considered suitable for predicting diabetes.
- Making machine learning model to predict diabetes risk
- Identification of the most important factor that contributes to the model's ability to predict diabetes.

Data Understanding

Dataset Detail

SOURCE

https://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset.

ATRIBUTES INFORMATION

Attribues	Values
Age	1.20-35, 2.36-45, 3.46-55,4.56-65, 6.above 65
Sex	1.Male, 2.Female
Polyuria	1.Yes, 2.No.
Polydipsia	1.Yes, 2.No.
sudden weight loss	1.Yes, 2.No.
weakness	1.Yes, 2.No.
Polyphagia	1.Yes, 2.No.
Genital thrush	1.Yes, 2.No.
visual blurring	1.Yes, 2.No.
Itching	1.Yes, 2.No.
Irritability	1.Yes, 2.No.
delayed healing	1.Yes, 2.No.
partial paresis	1.Yes, 2.No.
muscle stiffness	1.Yes, 2.No.
Alopecia	1.Yes, 2.No.
Obesity	1.Yes, 2.No.
Class	1.Positive, 2.Negative.

Data Information & Statistic Numerical

- From this information, dataset have 17 columns with 520 entries and data type from each column.
- Have 1 Numerical column (AGE) and 16 categorical column.
- The oldest person from dataset is 90 years old and the youngest is 16 years old

```
#getting an overall look over
df.describe()
```

	Age
count	520.000000
mean	48.028846
std	12.151466
min	16.000000
25%	39.000000
50%	47.500000
75%	57.000000
max	90.000000

```
[44] #checking the data-types and another way to chec
     df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  520 non-null    int64
 1   Gender               520 non-null    object
 2   Polyuria             520 non-null    object
 3   Polydipsia           520 non-null    object
 4   sudden weight loss   520 non-null    object
 5   weakness             520 non-null    object
 6   Polyphagia           520 non-null    object
 7   Genital thrush       520 non-null    object
 8   visual blurring      520 non-null    object
 9   Itching              520 non-null    object
 10  Irritability         520 non-null    object
 11  delayed healing      520 non-null    object
 12  partial paresis      520 non-null    object
 13  muscle stiffness     520 non-null    object
 14  Alopecia             520 non-null    object
 15  Obesity              520 non-null    object
 16  class                520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

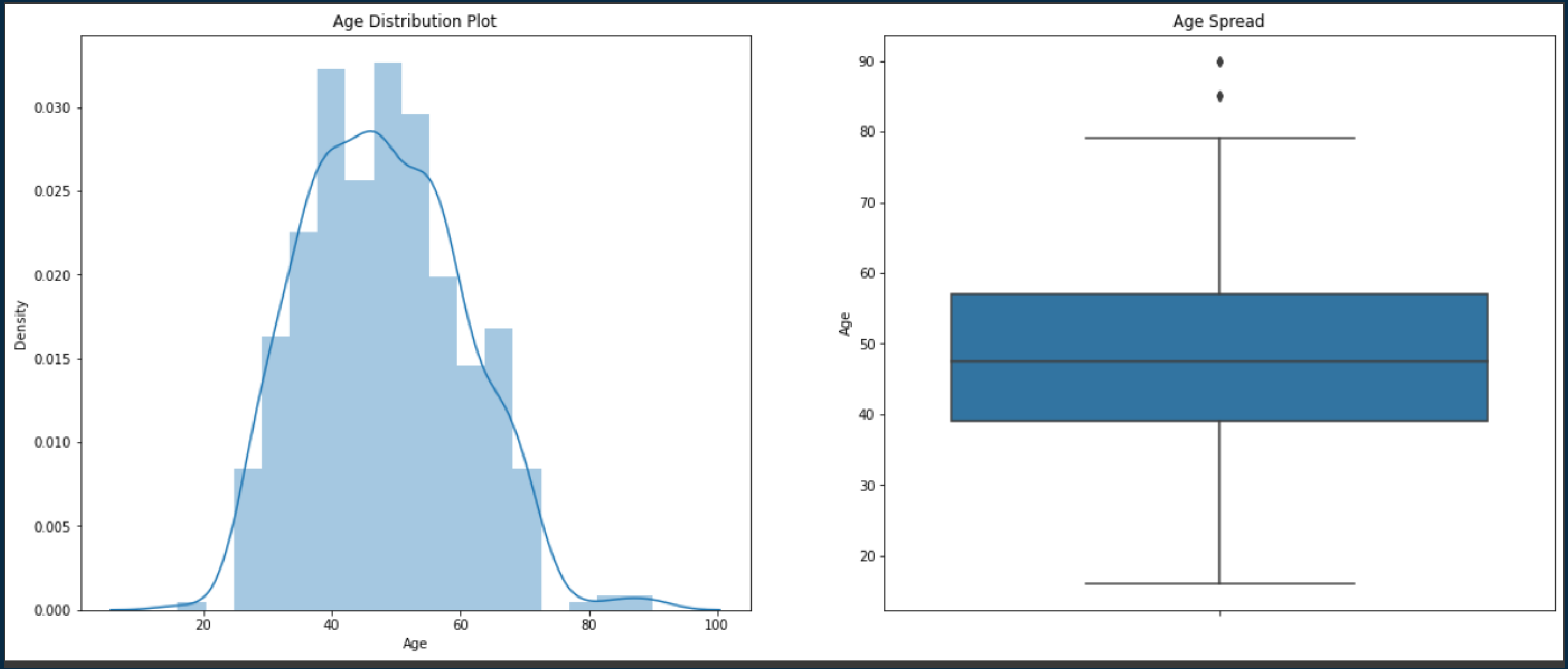
```
categorical = df.select_dtypes(exclude=[np.number])
categorical.columns

Index(['Gender', 'Polyuria', 'Polydipsia', 'sudden_weight_loss', 'weakness',
       'Polyphagia', 'Genital_thrush', 'visual_blurring', 'Itching',
       'Irritability', 'delayed_healing', 'partial_paresis',
       'muscle_stiffness', 'Alopecia', 'Obesity', 'class'],
      dtype='object')
```

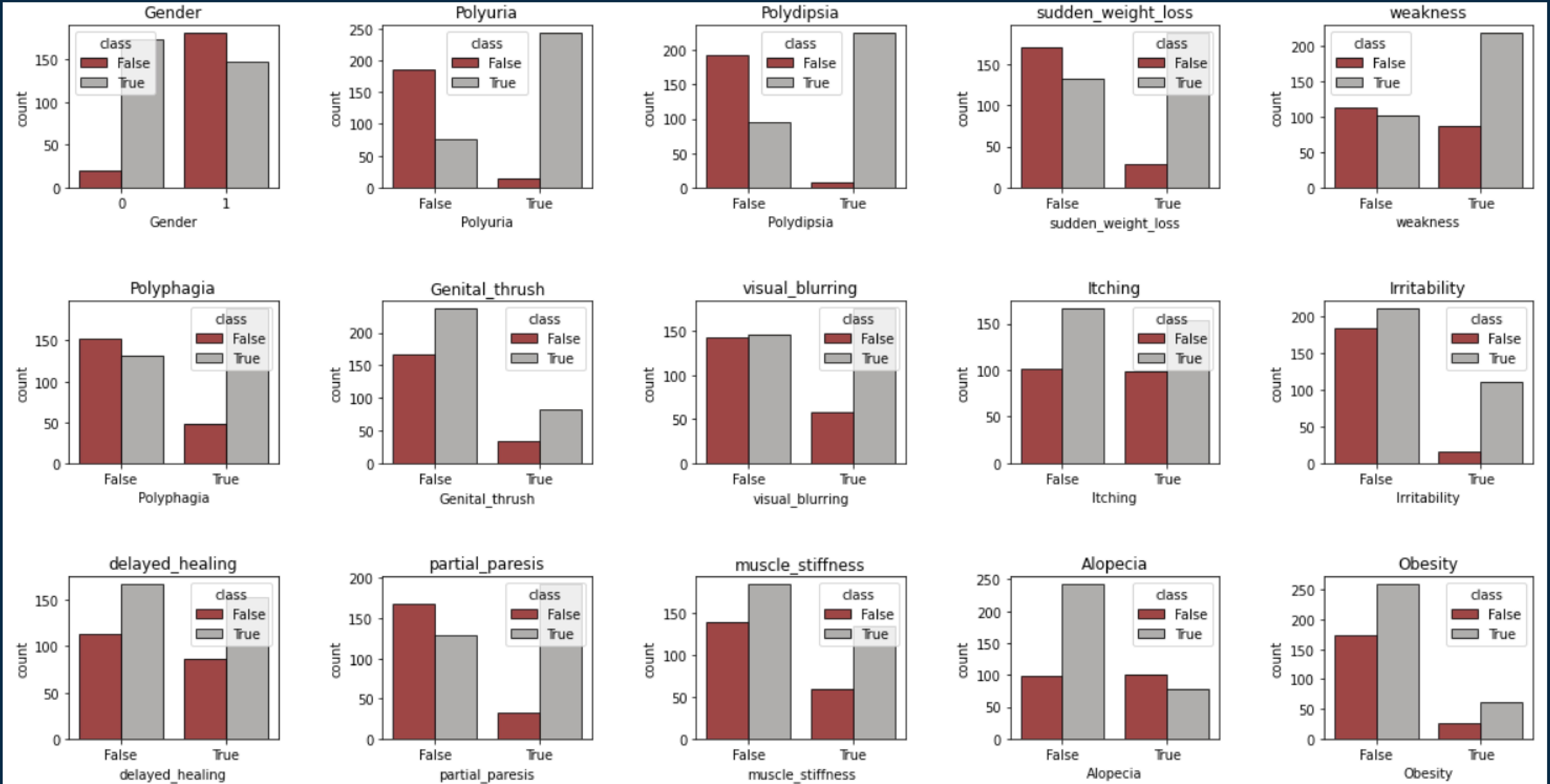
Exploratory Data Analysis
&
Data Visualization

Data Numerical Distribution

- Based on the graph above distribution of age has the highest density which at 45-55 years old. Box Plot show some ouliers, but we not remove it.



Data Categorical Bar Chart



Bar Chart Explaination

Based on the insights displayed, diabetes is largely related to the following factors:

- Diabetes is most prevalent among women, people who have polyuria, polydipsia, sudden weight loss, weakness, polyphagia, and partial paresis.
- Diabetes can also occur in people who do not have factors such as thrush, itching, irritability, delayed healing, muscle stiffness, alopecia, and obesity."

Data Preprocessing

Data Preprocessing

Data Preprocessing we do first is rename some columns to prevent errors from occurring

```
[94] df. rename(columns = {'sudden weight loss':'sudden_weight_loss',
                           'Genital thrush':'Genital_thrush',
                           'visual blurring':'visual_blurring',
                           'delayed healing':'delayed_healing',
                           'partial paresis':'partial_paresis',
                           'muscle stiffness':'muscle_stiffness'}, inplace = True)
```

Data Preprocessing

Modifying a majority of the contents in a categorical column, from Yes/No into boolean True/False using map() function.

Changing other column, gender and class too.

```
df['Polyuria'] = df['Polyuria'].map({'Yes':True,'No':False})
df['Polydipsia'] = df['Polydipsia'].map({'Yes':True,'No':False})
df['sudden_weight_loss'] = df['sudden_weight_loss'].map({'Yes':True,'No':False})
df['weakness'] = df['weakness'].map({'Yes':True,'No':False})
df['Polyphagia'] = df['Polyphagia'].map({'Yes':True,'No':False})
df['Genital_thrush'] = df['Genital_thrush'].map({'Yes':True,'No':False})
df['visual_blurring'] = df['visual_blurring'].map({'Yes':True,'No':False})
df['Itching'] = df['Itching'].map({'Yes':True,'No':False})
df['Irritability'] = df['Irritability'].map({'Yes':True,'No':False})
df['delayed_healing'] = df['delayed_healing'].map({'Yes':True,'No':False})
df['partial_paresis'] = df['partial_paresis'].map({'Yes':True,'No':False})
df['muscle_stiffness'] = df['muscle_stiffness'].map({'Yes':True,'No':False})
df['Alopecia'] = df['Alopecia'].map({'Yes':True,'No':False})
df['Obesity'] = df['Obesity'].map({'Yes':True,'No':False})
```

```
[107] df['Gender'].replace({'Female':0,'Male':1},inplace=True)
      df['class'] = df['class'].map({'Positive':True,'Negative':False})
```

Data Preprocessing

Split Data between categorical and numerical data, and than check null, duplicated, and unique value from data set.

```
categorical = df.select_dtypes(exclude=[np.number])
categorical.columns
```

```
Index(['Gender', 'Polyuria', 'Polydipsia', 'sudden_weight_loss', 'weakness',
       'Polyphagia', 'Genital_thrush', 'visual_blurring', 'Itching',
       'Irritability', 'delayed_healing', 'partial_paresis',
       'muscle_stiffness', 'Alopecia', 'Obesity', 'class'],
      dtype='object')
```

```
numerical = df.select_dtypes(include=[np.number])
numerical.columns
```

```
Index(['Age'], dtype='object')
```

```
[110] df.isnull().sum()
```

```
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
```

Data Modelling
&
Model Evaluation

Split Data & Data Shape

For Modelling process, we split data with ratio 80 data train :20 data test, and we check shape of data train and test

```
[63] X = df.drop(columns='class')
     y = df['class']
```

```
[64] from sklearn.model_selection import train_test_split,cross_validate
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

```
[65] print(f'X_train Shape: {(X_train.shape)}')
     print(f'y_train Shape: {(y_train.shape)}')
     print(f'X_test Shape: {(X_test.shape)}')
     print(f'y_test Shape: {(y_test.shape)}')
```

```
X_train Shape: (416, 16)
y_train Shape: (416,)
X_test Shape: (104, 16)
y_test Shape: (104,)
```

Decision Tree With Entropy

Decision Tree Classifier with criterion entropy

```
116] from sklearn.tree import DecisionTreeClassifier
     from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
     from sklearn.metrics import classification_report
     from sklearn.metrics import confusion_matrix, accuracy_score, make_scorer

     # Create Decision Tree classifer object
     clf = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=1)

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

     #Predict the response for test dataset
     y_pred_en = clf.predict(X_test)
     print(('Accuracy Entropy Model:'), accuracy_score(y_test,y_pred_en)*100)
```

```
Accuracy Entropy Model: 91.34615384615384
```

```
117] print('Training set score: {:.4f}'.format(clf.score(X_train, y_train)))
     print('Test set score: {:.4f}'.format(clf.score(X_test, y_test)))
```

```
Training set score: 0.8894
Test set score: 0.9135
```

```
[118] print(classification_report(y_test, y_pred_en))

      print("CONFUSION MATRIX")
      cnf_matrix=confusion_matrix(y_test, y_pred_en)
      print(cnf_matrix)
```

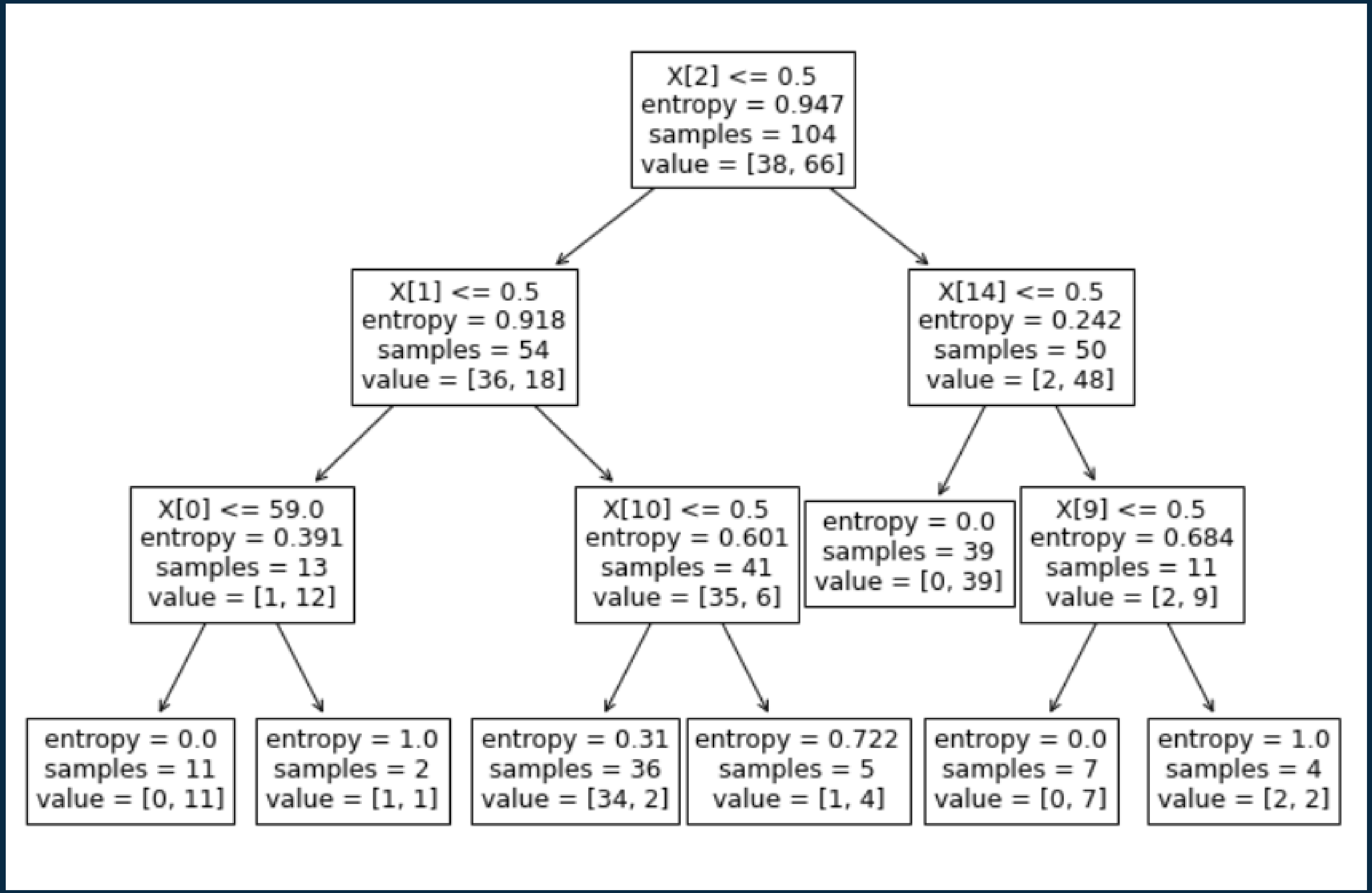
```
              precision    recall  f1-score   support

       False       0.94      0.82      0.87        38
        True       0.90      0.97      0.93        66

    accuracy                           0.91       104
   macro avg       0.92      0.89      0.90       104
weighted avg       0.92      0.91      0.91       104

CONFUSION MATRIX
[[31  7]
 [ 2 64]]
```

Tree With Entropy Visualization



Decision Tree With Gini Index

Decision Tree Classifier with criterion gini index

```
[170] clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1)
      # fit the model
      clf_gini.fit(X_train, y_train)
      y_pred_gini=clf_gini.predict(X_test)
      print(('Accuracy Model:'), accuracy_score(y_test,y_pred_gini)*100)
```

```
Accuracy Model: 92.3076923076923
```

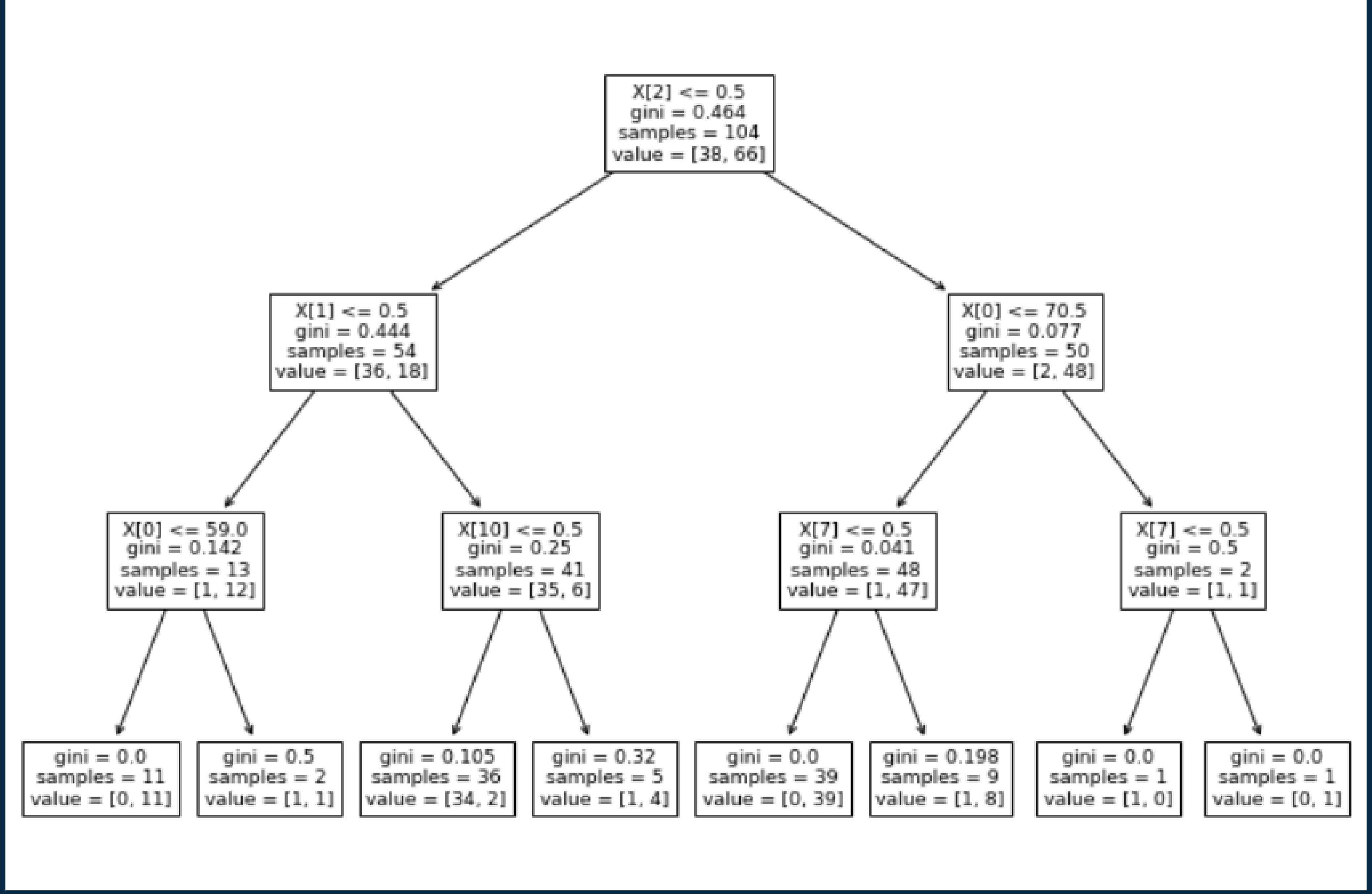
```
[171] print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))
      print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))
```

```
Training set score: 0.9038
Test set score: 0.9231
```

	precision	recall	f1-score	support
False	0.94	0.84	0.89	38
True	0.91	0.97	0.94	66
accuracy			0.92	104
macro avg	0.93	0.91	0.92	104
weighted avg	0.92	0.92	0.92	104

```
CONFUSION MATRIX
[[32  6]
 [ 2 64]]
```

Tree With Gini Index Visualization



DT with Cross Validation Result

Gini Index

Accuracy: 91.731%
Recall: 95.312%
Precision: 91.799%

Entropy

Accuracy: 90.192%
Recall: 95%
Precision: 89.941%

Random Forest Classifier (Without Huperparameter Tuning)

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=1)
rf_model.fit(X_train,y_train)
rf_pred=rf_model.predict(X_test)
print(('Accuracy Model:'), accuracy_score(y_test,rf_pred)*100)
```

```
Accuracy Model: 98.07692307692307
```

```
[38] print('Training set score: {:.4f}'.format(rf_model.score(X_train, y_train)))
     print('Test set score: {:.4f}'.format(rf_model.score(X_test, y_test)))
```

```
Training set score: 1.0000
Test set score: 0.9808
```

	precision	recall	f1-score	support
False	1.00	0.95	0.97	38
True	0.97	1.00	0.99	66
accuracy			0.98	104
macro avg	0.99	0.97	0.98	104
weighted avg	0.98	0.98	0.98	104

```
CONFUSION MATRIX
[[36  2]
 [ 0 66]]
```

With CV

Accuracy: 97.885%

Recall: 97.812%

Precision: 98.749%

Hyperparameter Tuning

For this case, we do Random Forest with hypertuning parameter, so we use get the best parameter using GridSearchCV, then fit to the model. Best parameter we use is {'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 500}

```
[41] param_grid = {
         'n_estimators': [100,200, 500,1000],
         'max_features': ['auto', 'sqrt', 'log2'],
         'max_depth' : [4,5,6,7,8],
         'criterion' :['gini', 'entropy']
     }
```

```
[42] from sklearn.model_selection import GridSearchCV
     CV_rf = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv= 10)
     CV_rf.fit(X_train, y_train)

     GridSearchCV(cv=10, estimator=RandomForestClassifier(random_state=1),
                  param_grid={'criterion': ['gini', 'entropy'],
                              'max_depth': [4, 5, 6, 7, 8],
                              'max_features': ['auto', 'sqrt', 'log2'],
                              'n_estimators': [100, 200, 500, 1000]})
```

```
CV_rf.best_params_

{'criterion': 'gini',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 500}
```

Random Forest Classifier (With Huperparameter Tuning)

After that, we make new model with best parameter and predict data test, get score from that model, and show classification report & Confusion Matrix.

```
[44] rf2 = RandomForestClassifier(n_estimators = 500, max_depth = 8, max_features = 'auto', criterion = 'gini').fit(X_train, y_train)
     rf2_pred=rf2.predict(X_test)
     print(('Accuracy Model:'), accuracy_score(y_test,rf2_pred)*100)

     Accuracy Model: 98.07692307692307

[45] print(classification_report(y_test, rf2_pred))

     print("CONFUSION MATRIX")
     cnf_matrix=confusion_matrix(y_test, rf2_pred)
     print(cnf_matrix)

                   precision    recall  f1-score   support

            False       1.00      0.95      0.97        38
             True       0.97      1.00      0.99        66

         accuracy                           0.98       104
        macro avg       0.99      0.97      0.98       104
     weighted avg       0.98      0.98      0.98       104

     CONFUSION MATRIX
     [[36  2]
      [ 0 66]]
```

Feature Importances

find importance value in every feature using feature_importances_ function from Random Forest Model. Polydipsia is the most importance feature in this dataset and least feature is obesity

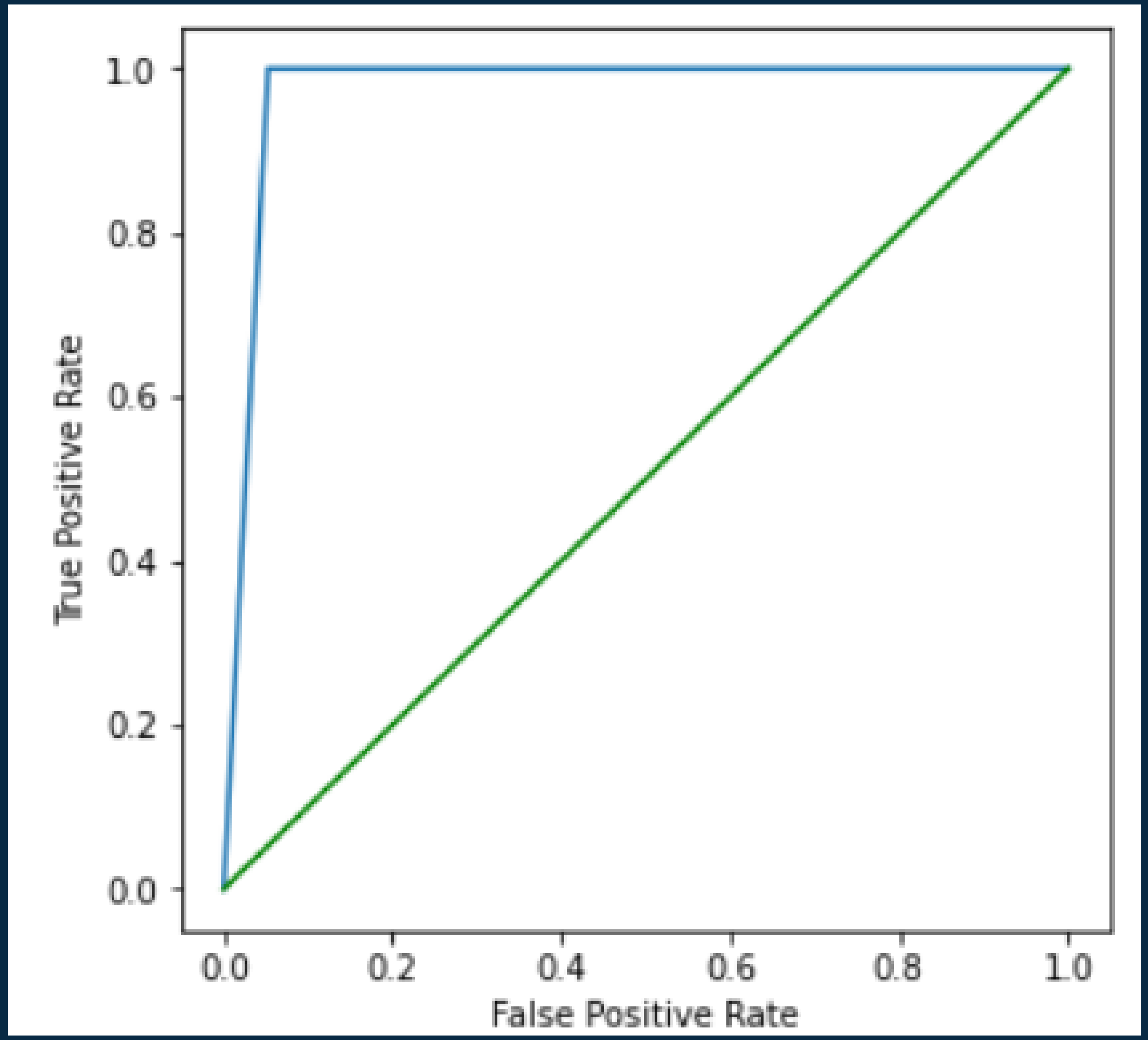
```
[46] f_list= list(X.columns)
     f_importance=pd.Series(rf2.feature_importances_, index=f_list).sort_values(ascending=False)

     print(f_importance)
```

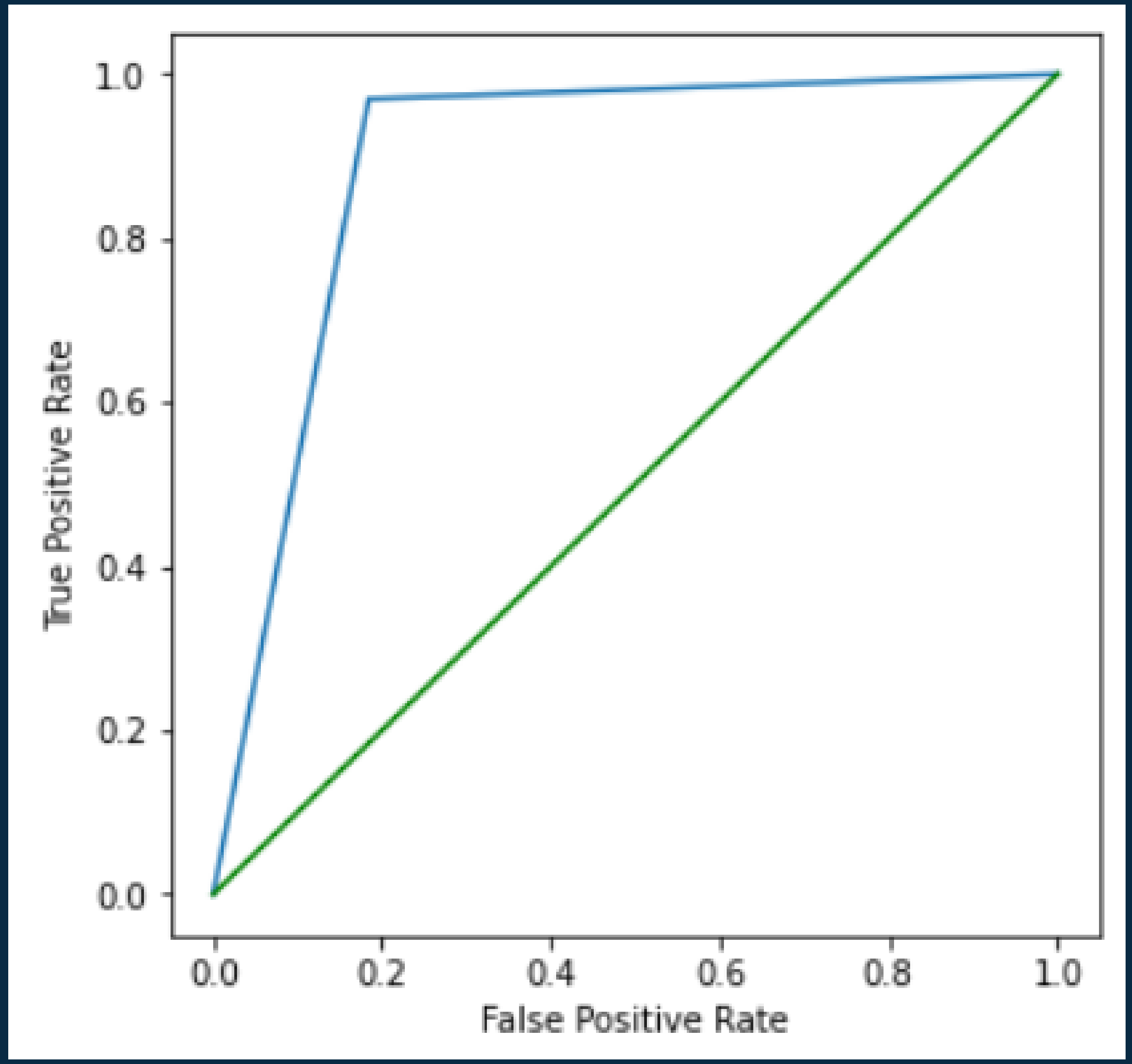
```
Polydipsia            0.239237
Polyuria              0.196680
Age                   0.085708
Gender                0.082098
sudden_weight_loss    0.061715
partial_paresis       0.049291
Irritability          0.040903
Alopecia              0.036875
Polyphagia            0.033842
visual_blurring       0.031464
delayed_healing       0.031138
Itching               0.029245
muscle_stiffness      0.022396
Genital_thrush        0.021670
weakness              0.020409
Obesity               0.017328
dtype: float64
```

ROC AUC Curve

Random Forest



DT Entropy



Conclusion

- THE RANDOM FOREST MODEL PRODUCED THE HIGHEST ACCURACY COMPARED TO TWO DECISION TREE MODELS, WITH 98,07% ACCURACY.
- THE PERFORMANCE OF THE RANDOM FOREST MODEL IS THE SAME WHETHER IT IS WITH OR WITHOUT HYPERPARAMETER TUNING.
- POLYDIPSIA IS THE MOST IMPORTANCE FEATURE.
- AS THE DATA MINING METHODS, TECHNIQUES AND TOOLS ARE BECOMING MORE PROMISING TO PREDICT DIABETES AND EVENTUALLY NUMBER OF PATIENTS REDUCE THE TREATMENT COST, ITS ROLE IN THIS MEDICAL HEALTH CARE IS UNDENIABLE.

Thank you!

EMAIL

randayandika1@gmail.com

LINKEDIN

linkedin.com/in/muhammad-randa-yandika

GITHUB

github.com/randayandika