DIABETES PREDICTION SYSTEM

Python20 - Final Project

Ho Linh Chi

CONTENTS

INTRODUCTION

- Hồ Linh Chi
- Email: chiholinh2001@gmail.com
- Linkedin: https://www.linkedin.com/in/chiholinh/
- Course: Python for Data Science - Python20

PROJECT

- **Diabetes** is a chronic (long-lasting) health condition that affects how your body turns food into energy. It affects the quality of health and life.
- Diabetes has **difficult-to-recognize symptoms**.

=> We need to predict the disease as soon as possible by **Diabetes Prediction System**.

INPUT & OUTPUT

INPUT

- Pregnancies
- Glucose
- Blood Pressure (mm Hg)
- Skin Thickness (mm)
- Insulin
- BMI
- Diabetes Pedigree Function
- Age

OUTPUT

Have diabetes or not

01 This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

02 Link download:
https://www.kaggle.com/datasets/mathchi/diabetes-data-set

DATASET

Diabetes

```
[ ] dataset.head()
```

	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

```
[ ] dataset.shape
```

```
(768, 9)
```

```
[ ] dataset.dtypes
```

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

DATASET

Diabetes

```
[ ] dataset.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

```
[ ] dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

DATASET

Diabetes

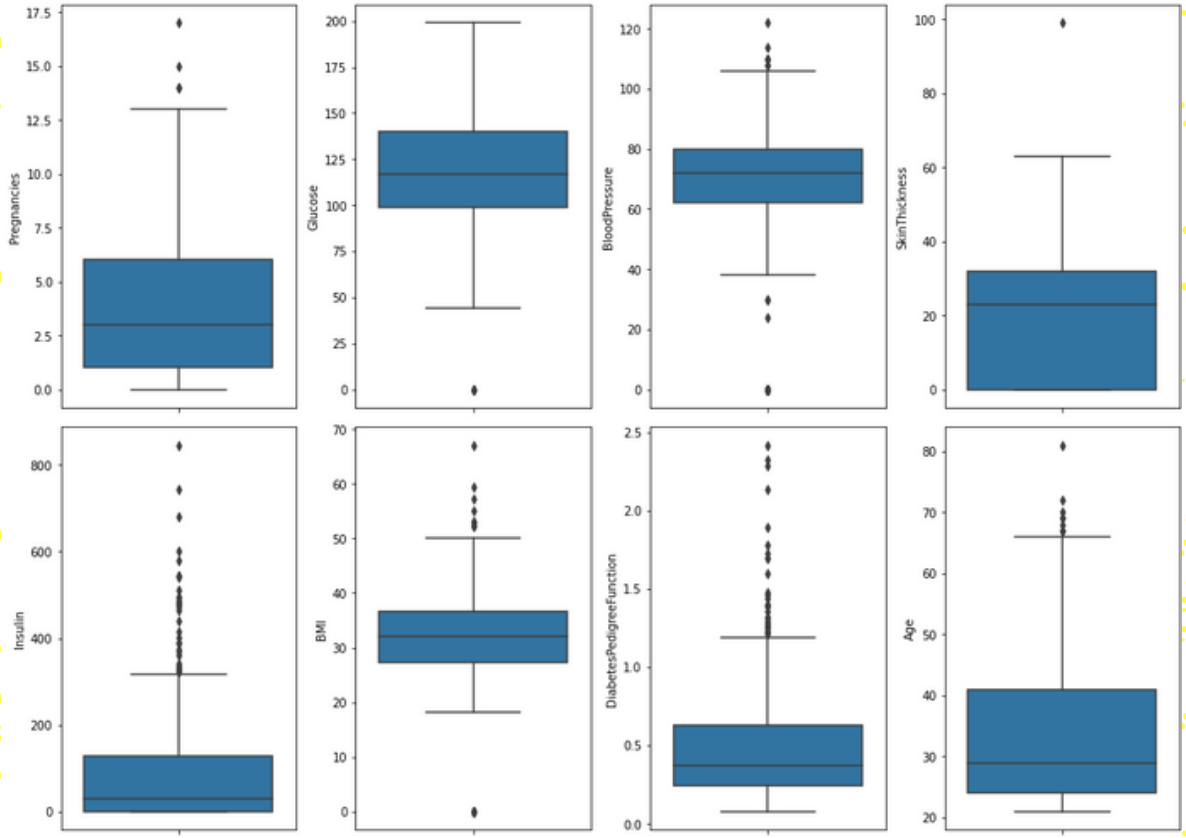
```
[96] dataset['Outcome'].value_counts()

     0    500
     1    268
     Name: Outcome, dtype: int64
```

```
[97] dataset.groupby('Outcome').mean()
```

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734	31.190000
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500	37.067164

DATASET

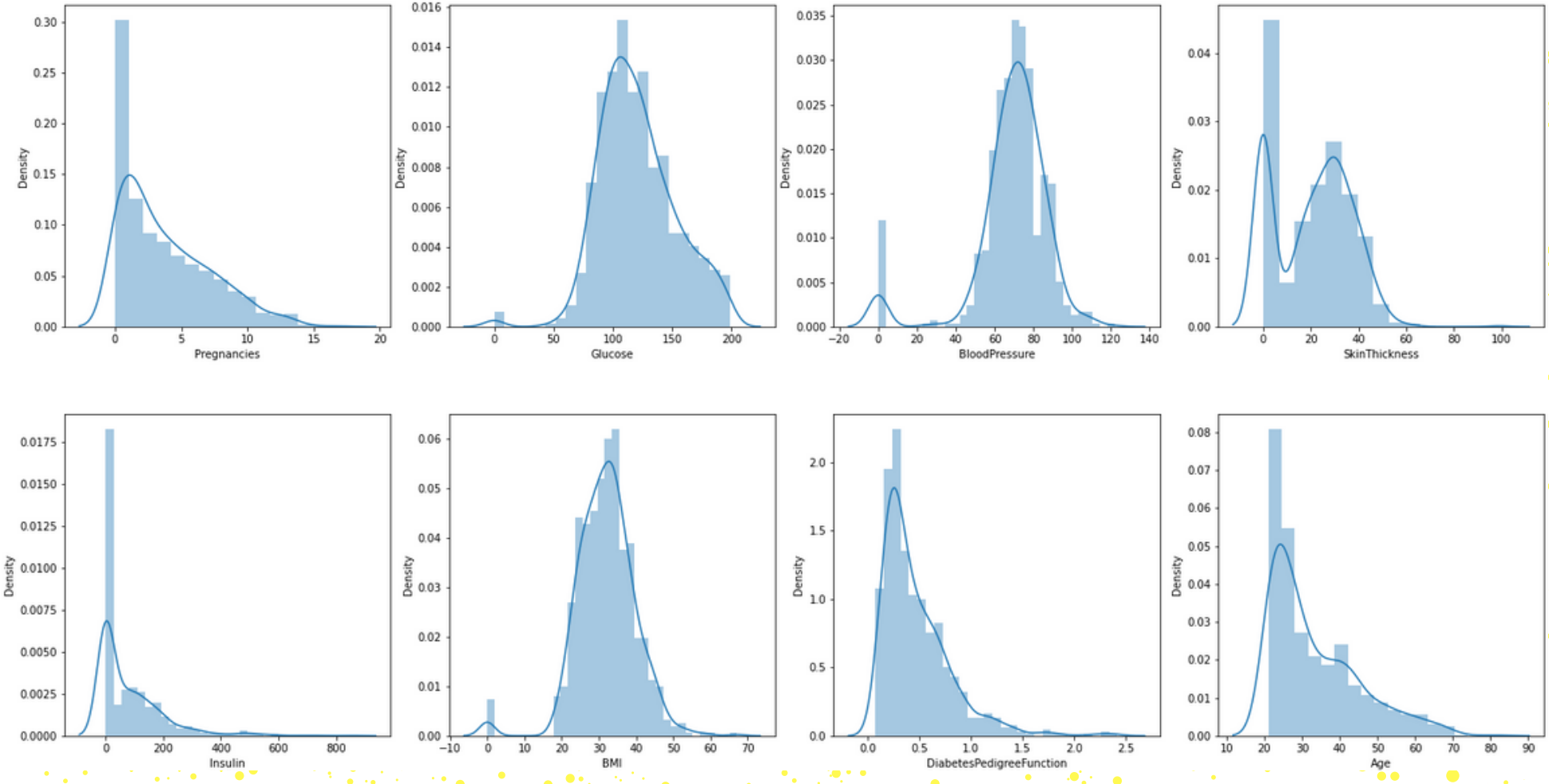
Diabetes




```
Column Pregnancies outliers = 0.52%
Column Glucose outliers = 0.65%
Column BloodPressure outliers = 5.86%
Column SkinThickness outliers = 0.13%
Column Insulin outliers = 4.43%
Column BMI outliers = 2.47%
Column DiabetesPedigreeFunction outliers = 3.78%
Column Age outliers = 1.17%
```

DATASET

Diabetes



DATASET

Diabetes

DATA PRE-PROCESSING

Upsampling dataset

BEFORE

```
6] dataset['Outcome'].value_counts()

0    500
1    268
Name: Outcome, dtype: int64
```

AFTER

```
] #upsampling
  df_majority = dataset[dataset['Outcome']==0]
  df_minority = dataset[dataset['Outcome']==1]
  df_minority_upsampled = resample(df_minority,
                                   replace=True,
                                   n_samples=500)
  dataset = pd.concat([df_majority, df_minority_upsampled])
  dataset['Outcome'].value_counts()

  0    500
  1    500
  Name: Outcome, dtype: int64
```

DATA PRE-PROCESSING

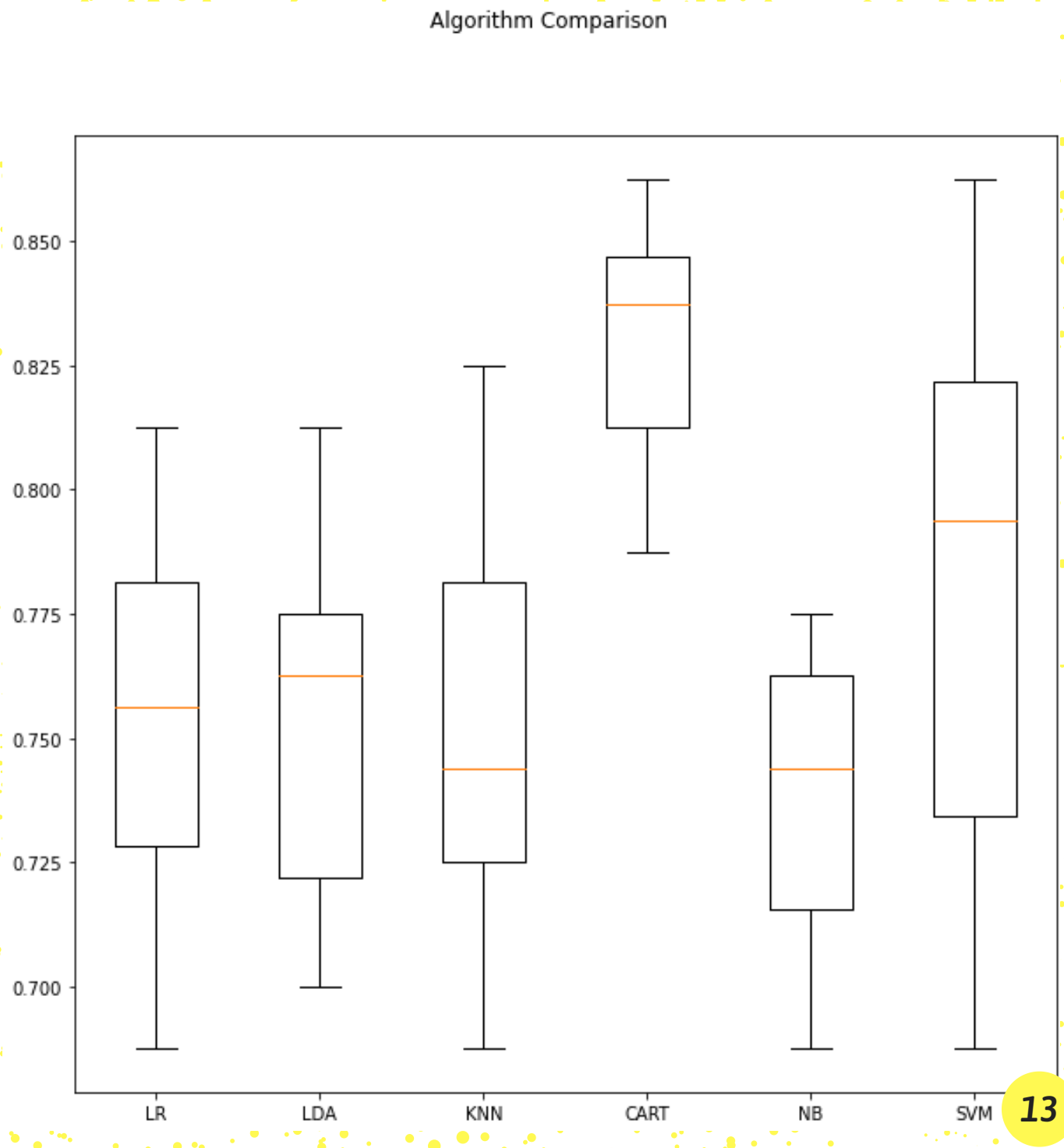
Data Standardization

```
#Data standardization
sc = StandardScaler()
sc.fit(X)
standardized_data = sc.transform(X)
print(standardized_data)
```

```
[[-0.86282741 -1.18388367 -0.13918399 ... -0.78929528 -0.41071698
  -0.27194763]
 [-0.86282741 -1.06515396 -0.13918399 ... -0.59062047 -0.96124785
  -1.12766704]
 [ 0.23456834 -0.26372838  0.25706936 ... -0.92174515 -0.85951932
  -0.35751957]
 ...
 [-1.13717635  0.09246077 -0.04012065 ... -0.27274078  2.70097928
  -0.78537927]
 [-1.13717635 -0.02626895  0.05894269 ... -0.68333538 -0.70094249
   0.15591208]
 [ 0.23456834 -0.26372838  0.25706936 ... -0.03433101  0.51381585
   0.07034014]]
```

COMPARE & CHOOSE MODELS

LR: 0.752500 (0.042500)
LDA: 0.756250 (0.039231)
KNN: 0.755000 (0.043012)
CART: 0.830000 (0.024495)
NB: 0.738750 (0.029817)
SVM: 0.780000 (0.056513)

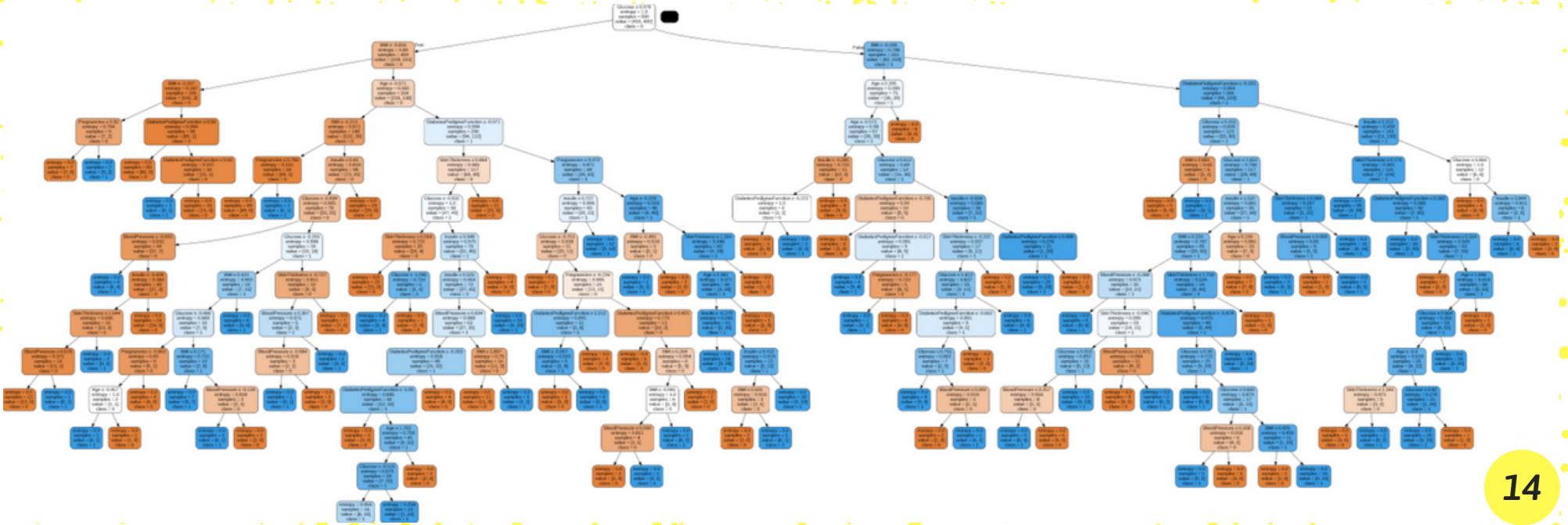


COMPARE & CHOOSE MODELS

Decision Trees Model

```
clf = DecisionTreeClassifier()
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.82
```



MODEL OPTIMIZATION

Decision Tree

```
pipe = Pipeline(steps=[('sc', sc),
                       ('dec_tree', clf)])
n_components = list(range(1,X.shape[1]+1,1))
criterion = ['gini', 'entropy']
max_depth = [2, 4, 6, 8, 10, 12, 13, 14, 15, 16, 17, 18]
splitter = ['best', 'random']
min_samples_split = [1, 2, 3, 4]
parameters = dict(dec_tree__criterion=criterion,
                  dec_tree__max_depth=max_depth,
                  dec_tree__splitter=splitter)
clf_GS = GridSearchCV(pipe, parameters)
clf_GS.fit(X, y)
print('Best Criterion:', clf_GS.best_estimator_.get_params()['dec_tree__criterion'])
print('Best max_depth:', clf_GS.best_estimator_.get_params()['dec_tree__max_depth'])
print('Best splitter:', clf_GS.best_estimator_.get_params()['dec_tree__splitter'])
print(); print(clf_GS.best_estimator_.get_params()['dec_tree'])
```

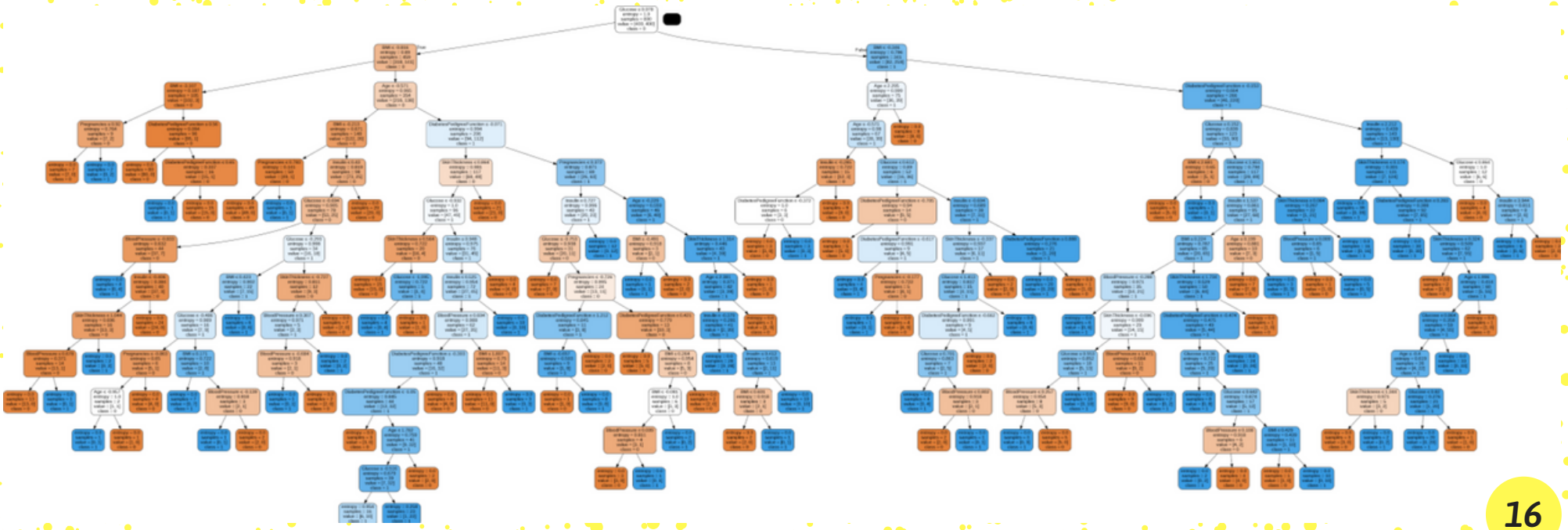
```
Best Criterion: entropy
Best max_depth: 13
Best splitter: best

DecisionTreeClassifier(criterion='entropy', max_depth=13)
```

MODEL OPTIMIZATION

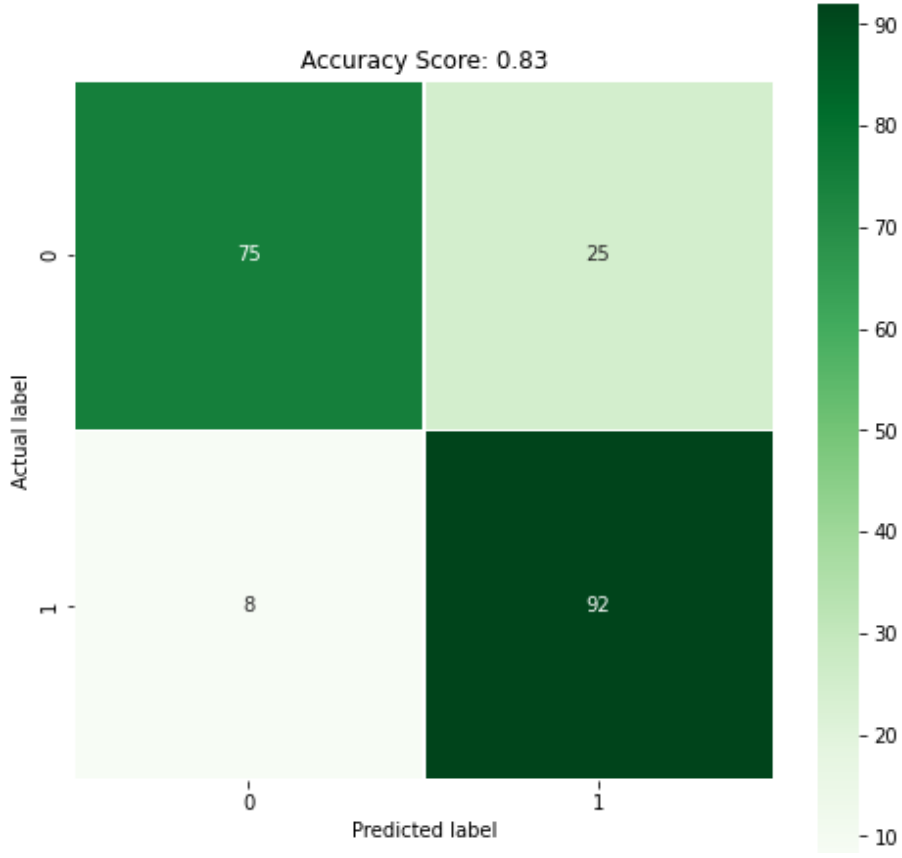
Decision Tree

```
clf = DecisionTreeClassifier(criterion='entropy', max_depth=13)
clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.835
```

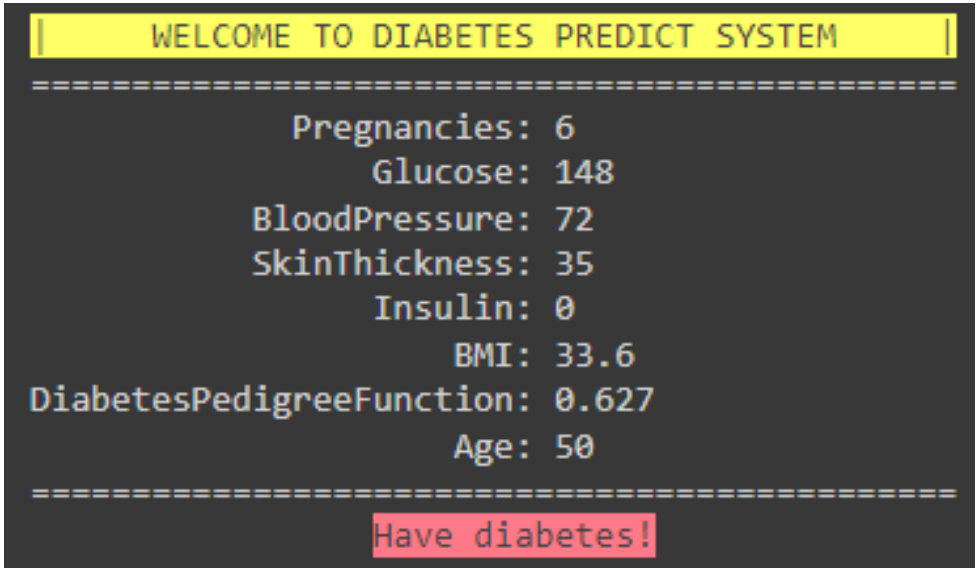


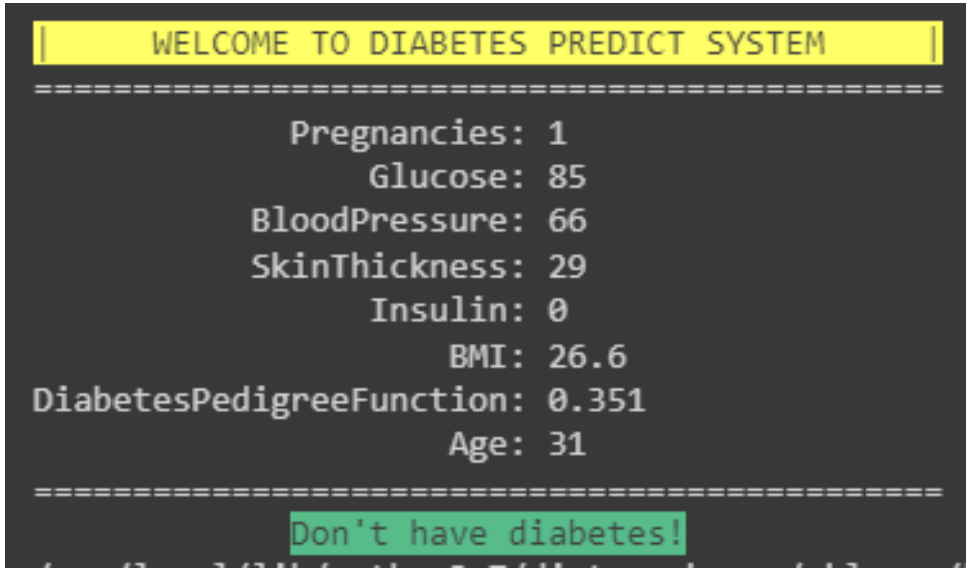
REVIEW



	precision	recall	f1-score	support
0	0.90	0.75	0.82	100
1	0.79	0.92	0.85	100
accuracy			0.83	200
macro avg	0.84	0.83	0.83	200
weighted avg	0.84	0.83	0.83	200

DIABETES PREDICTION SYSTEM





THANK YOU!

Ho Linh Chi

Diabetes Prediction System