DIABETES PREDICTION PROJECT

FOGUE KAMGA DILANE | DATA SCIENTIST

AGENDA

- ❑ Concept Study
- ❑ Data Preparation
- ❑ EDA
- ❑ Model Building
- ❑ Communicate Results
- ❑ Operationalize

Concept Study

In this project, we will be predicting that whether the patient has diabetes or not on the basis of the features(criteria) we will provide to our machine learning model.

Criteria

Pregnancies

Glucose

Blood Pressure

Skin Thickness

Insulin

BMI

Diabetes Pedigree Function

Age

Connect to App

information Submission

Prediction

Concept Study

- Importing libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import metrics
```

Data Preparation

❑ Read the dataset

```
df = pd.read_csv('diabetes.csv')
```

❑ Display the 5 first rows

```
df.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

Data Preparation

❑ Display the dimension of the dataset

```
df.shape
```

```
(768, 9)
```

❑ Information about the dataset

```
df.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
```

Data Preparation

❑ Describe the dataset to know more about the it

```
df.describe().T
```

	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

Data Preparation

- Check the number of missing values our dataset has

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

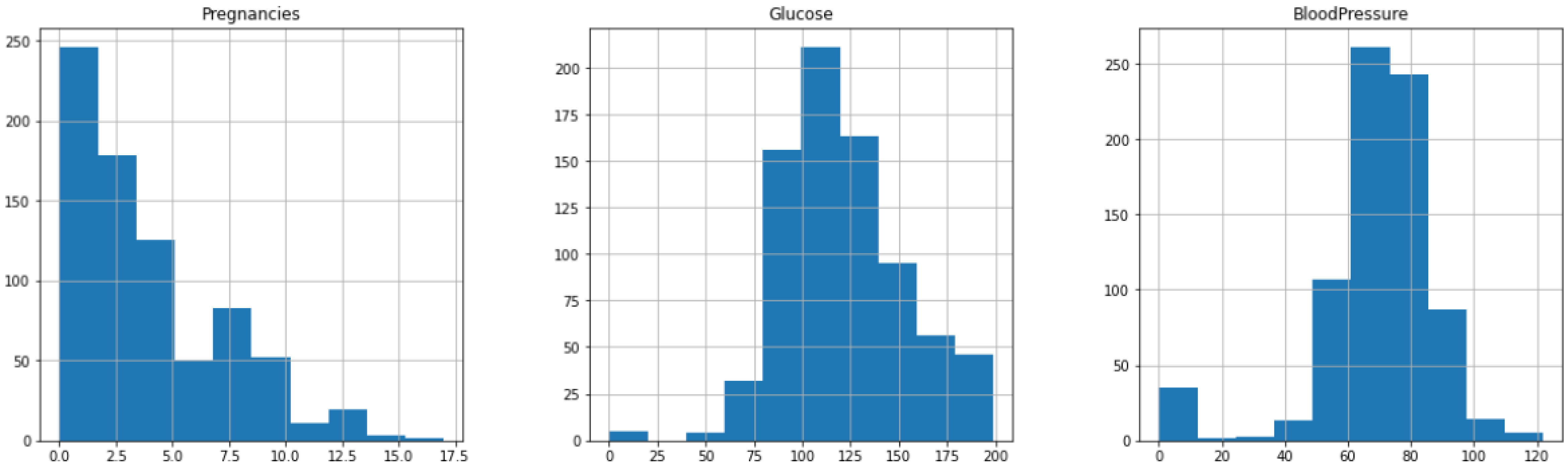
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

We now get is that there are no missing values but that is actually not a true story as in this particular dataset all the missing values were given the outlier(0) as a value which is not good for the authenticity of the dataset. So we will handle the outliers (0).

Data Preparation

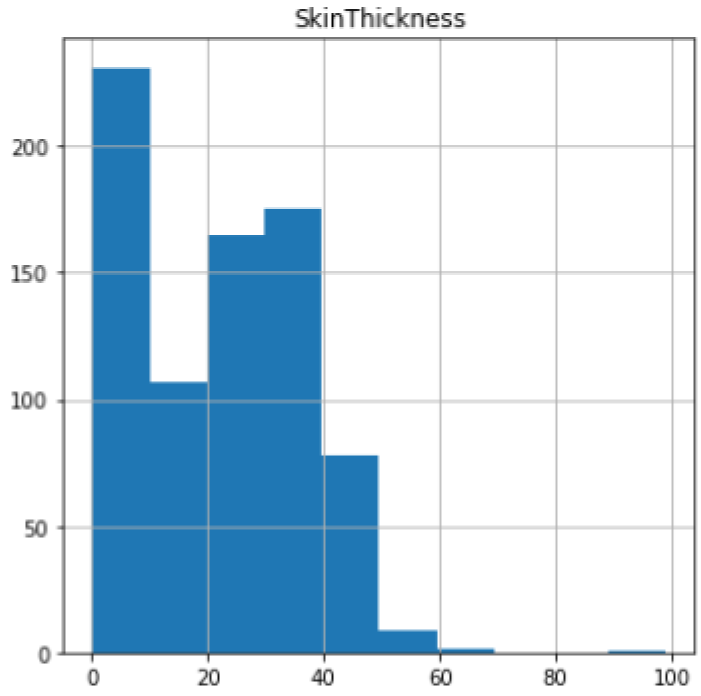
❑ Plotting the data distribution plots

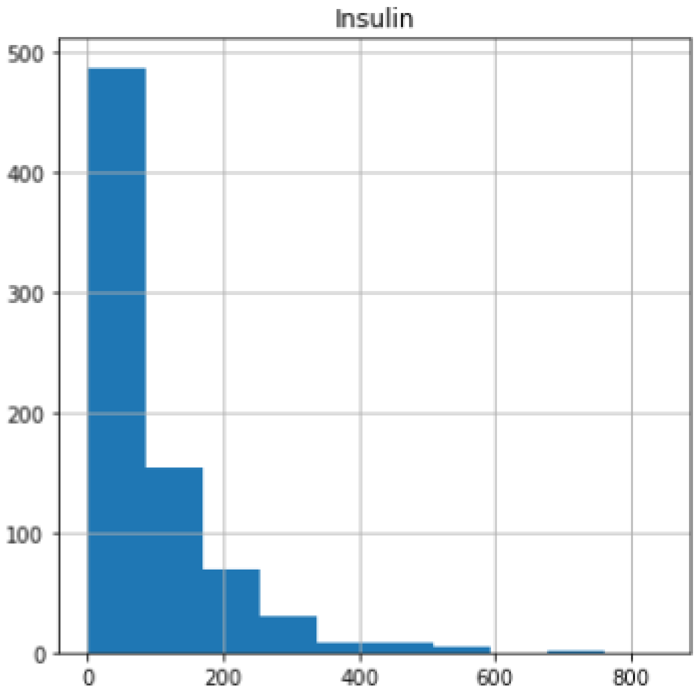
```
p = df.hist(figsize = (20,20))
```

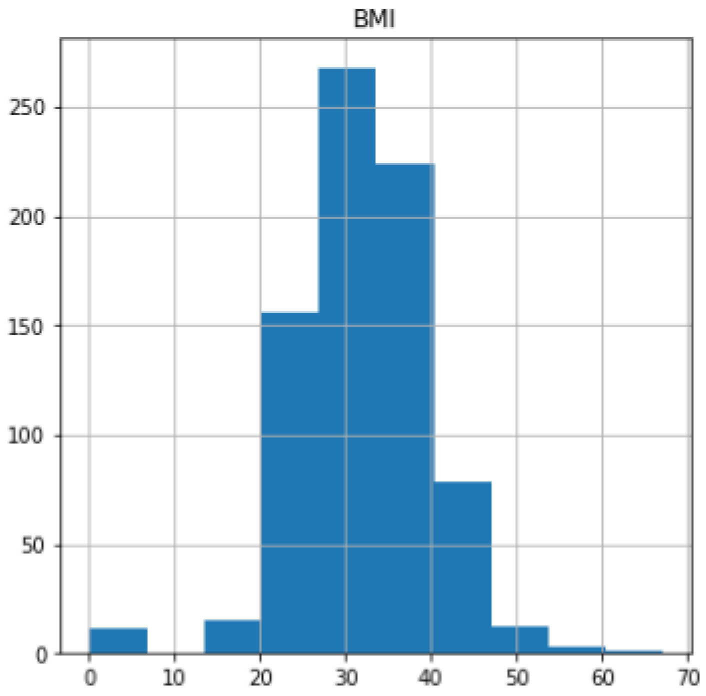


Data Preparation

❑ Plotting the data distribution plots

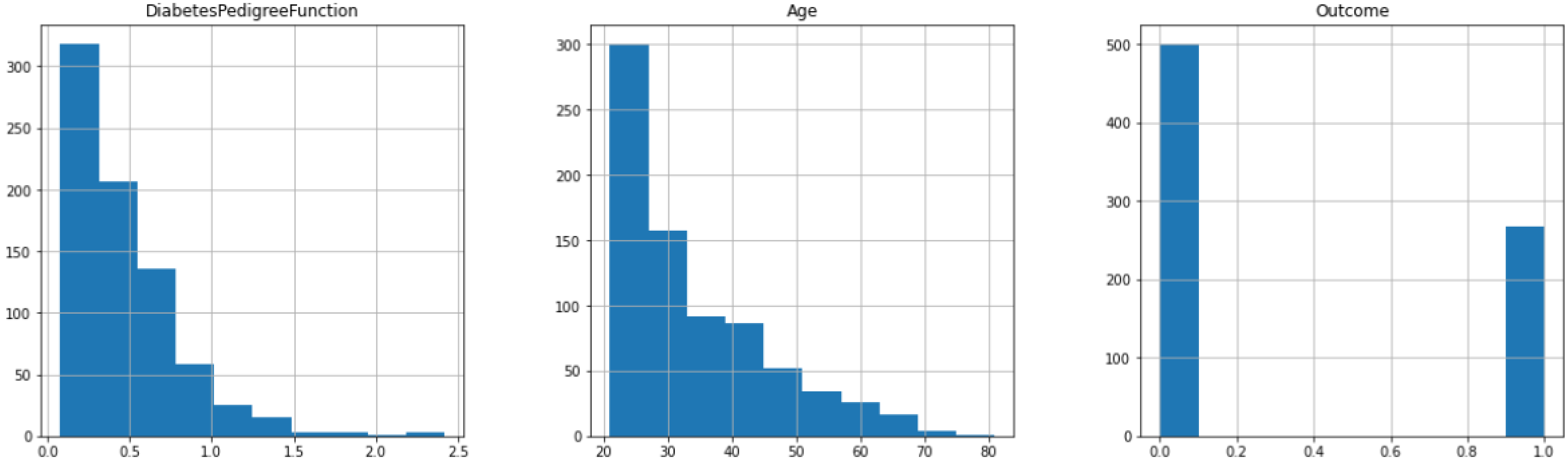






Data Preparation

❑ Plotting the data distribution plots



Data Preparation

❑ Handling the outliers

```
# Replace the 0 values with the NAN
df[
    ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
]  =
df[
    [['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

```
# check the number of missing values
df.isnull().sum()
```

```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

Data Preparation

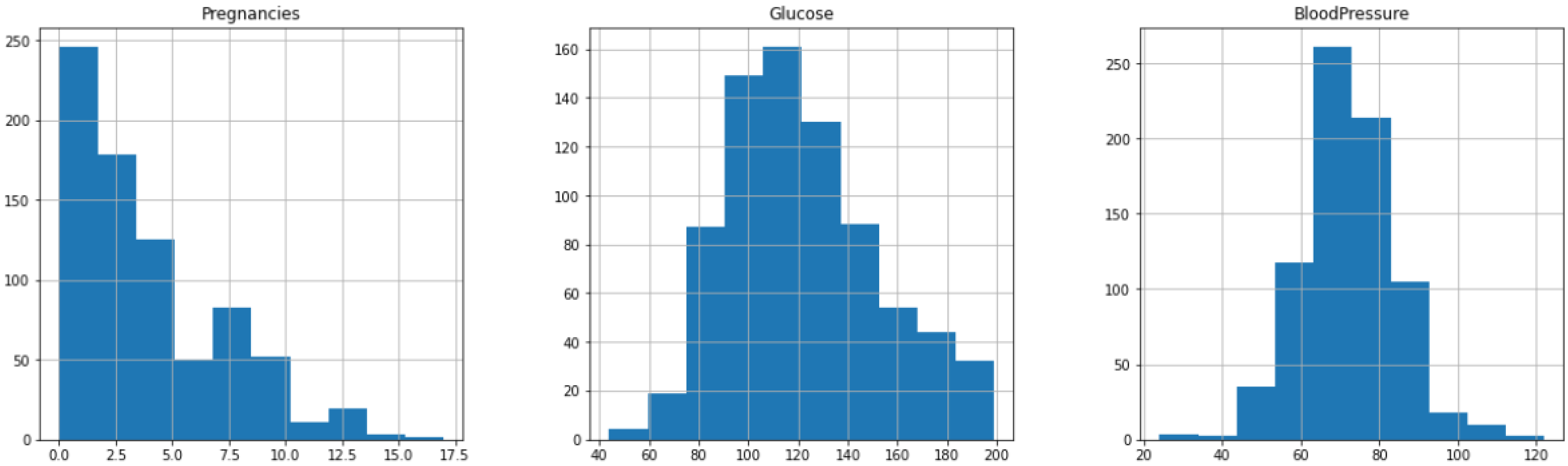
❑ Handling the outliers

```
# Replace the missing values
df['Glucose'].fillna(df['Glucose'].mean(), inplace = True)
df['BloodPressure'].fillna(df['BloodPressure'].mean(), inplace = True)
df['SkinThickness'].fillna(df['SkinThickness'].median(), inplace = True)
df['Insulin'].fillna(df['Insulin'].median(), inplace = True)
df['BMI'].fillna(df['BMI'].median(), inplace = True)
```

Data Preparation

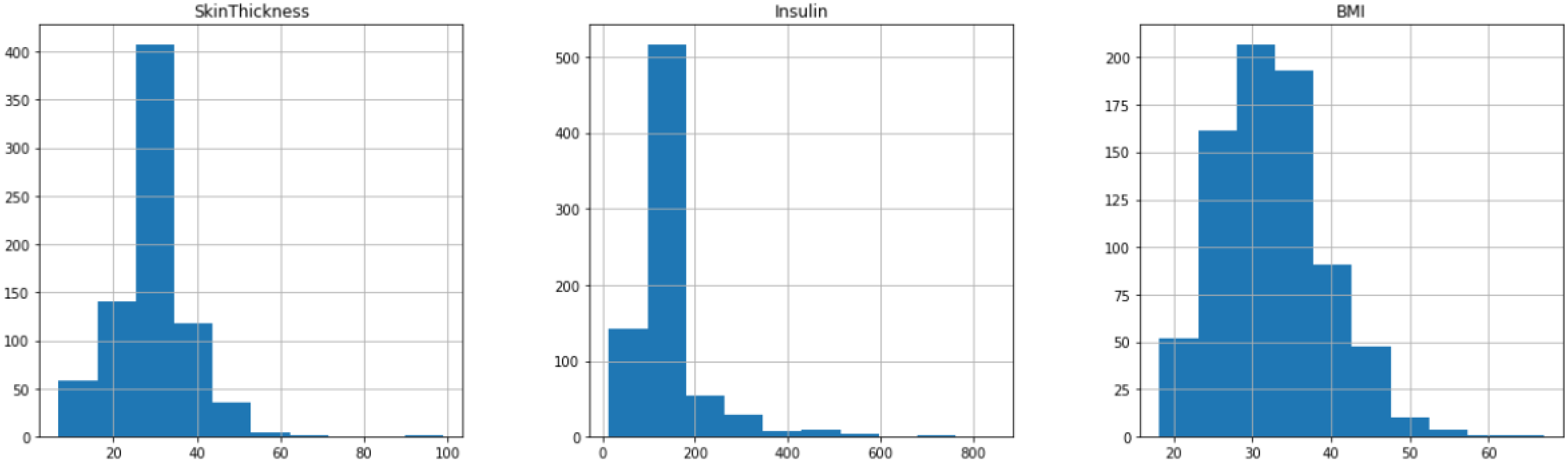
❑ Plotting the data distribution plots

```
p = df.hist(figsize = (20,20))
```



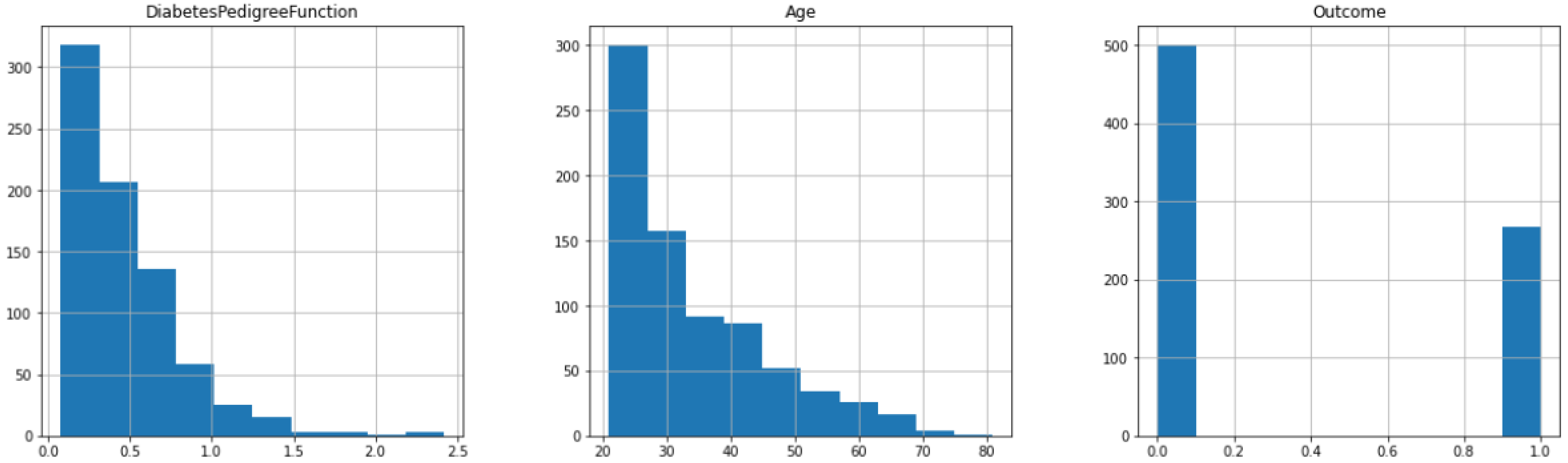
Data Preparation

❑ Plotting the data distribution plots



Data Preparation

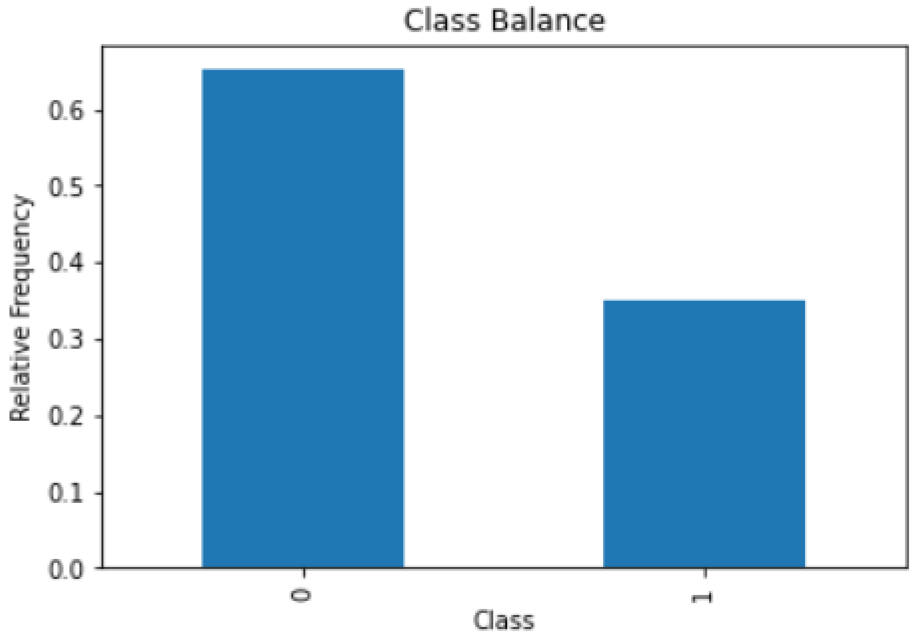
❑ Plotting the data distribution plots



EDA

❑ Check that how well our outcome column is balanced

```
# Plot value counts of `"Outcome"`
df["Outcome"].value_counts(normalize=True).plot(
    kind="bar", xlabel="Class", ylabel="Relative Frequency", title="Class Balance"
);
```



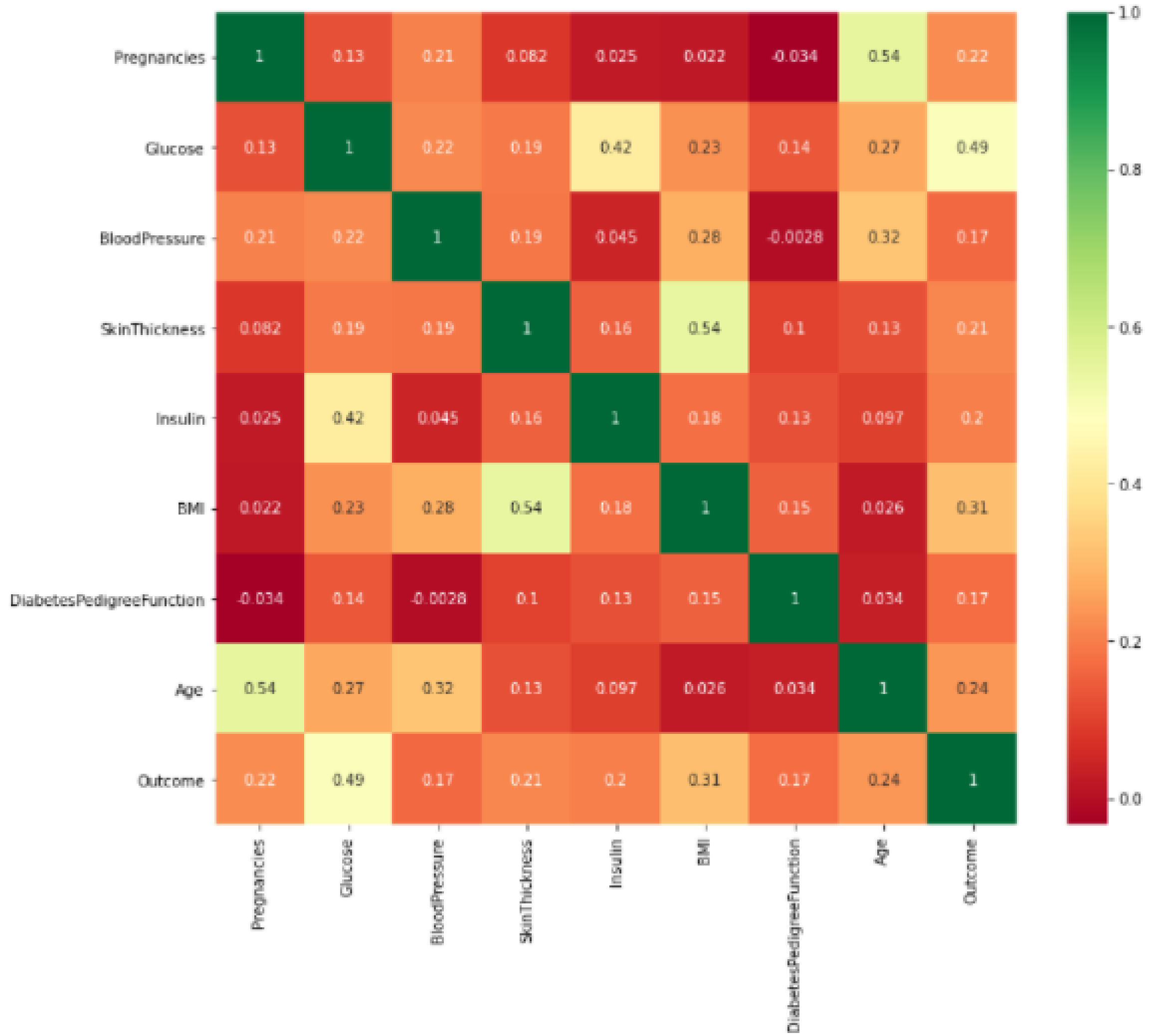
EDA

❑ Correlation between all the features

```
plt.figure(figsize=(12,10))
p = sns.heatmap(df.corr(), annot=True,cmap ='RdYlGn')
```

EDA

❑ Correlation between all the features



Model Building

Algorithms Used in this Project

- ❑ Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

- ❑ Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

- ❑ Support Vector Machine

```
from sklearn.svm import SVC
```

Model Building | Split Data

```python
# Create your feature matrix X and target vector y
target = "Outcome"
X = df.drop(columns=target)
y = df[target]
```

```python
# Features Scaling
sc = StandardScaler()
X = sc.fit_transform(X)
print(X)
```

```python
# Divide the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (614, 8)
y_train shape: (614,)
X_test shape: (154, 8)
y_test shape: (154,)
```

Model Building | Decision Tree

```
# Instanciate the model
DTClassifier = DecisionTreeClassifier()
```

```
# Fit the model
DTClassifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
# Prediction and accuracy
predictions = DTClassifier.predict(X_test)
print("Accuracy Score =", format(accuracy_score(y_test,predictions)))
```

```
Accuracy Score = 0.7662337662337663
```

Model Building | Random Forest

```
# Instanciate the model
RFClassifier = RandomForestClassifier(n_estimators=600)
```

```
# Fit the model
RFClassifier.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=600)
```

```
# Prediction and accuracy
predictions = RFClassifier.predict(X_test)
print("Accuracy_Score =", format(accuracy_score(y_test, predictions)))
```

```
Accuracy_Score = 0.7402597402597403
```

Model Building | Support Vector Machine

```
# Instanciate the model
svClassifier = SVC()
```

```
# Fit the model
svClassifier.fit(X_train, y_train)
```

```
SVC()
```

```
# Prediction and accuracy
svc_pred = svc_model.predict(X_test)
print("Accuracy Score =", format(accuracy_score(y_test, svc_pred)))
```

```
Accuracy Score = 0.7272727272727273
```

Communicate Results

❑ Decision Tree

```
# Display the confusion matrix
cm = confusion_matrix(predictions2, y_test)
cm
```

```
array([[79, 16],
       [20, 39]], dtype=int64)
```

❑ Random Forest

```
# Display the confusion matrix
cm = confusion_matrix(predictions, y_test)
cm
```

```
array([[78, 19],
       [21, 36]], dtype=int64)
```

❑ Support Vector Machine

```
# Display the confusion matrix
cm = confusion_matrix(svc_pred, y_test)
cm
```

```
array([[81, 24],
       [18, 31]], dtype=int64)
```

Communicate Results

Therefore Decision Tree is the best model for this prediction since it has an accuracy_score of 0.76

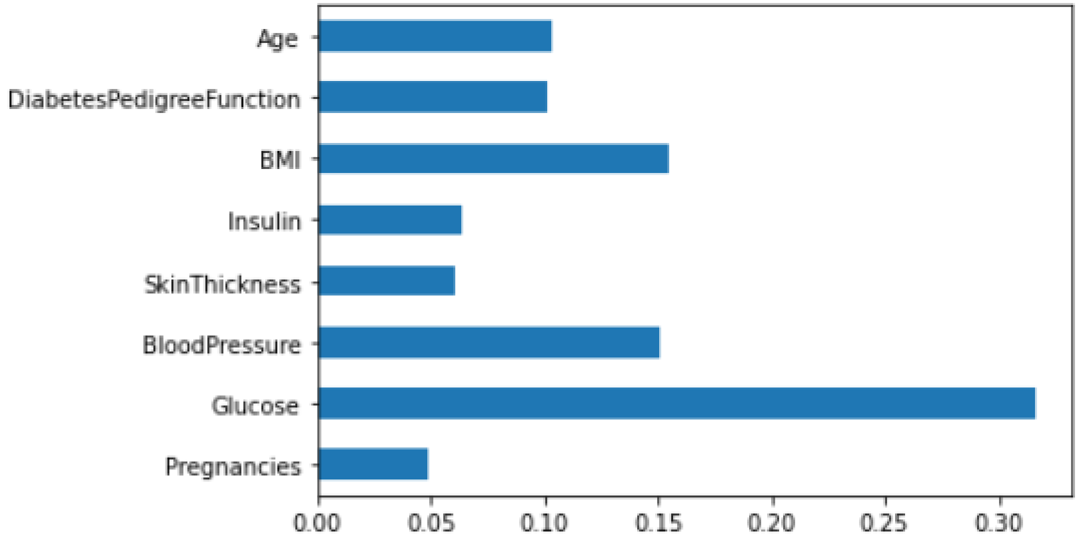
❑ Features importances

```
DTClassifier.feature_importances_
```

```
array([0.04885521, 0.31590208, 0.15113023, 0.0606083 , 0.06355162,
       0.15468574, 0.1017383 , 0.10352852])
```

```
(pd.Series(DTClassifier.feature_importances_, index=df.columns[:8]).plot(kind='barh'))
```

```
<AxesSubplot:>
```



Communicate Results

Here from the above graph, it is clearly visible that Glucose as a feature is the most important in this dataset.

- ❑ Saving the model

```
filename = 'finalized_model.pkl'
pickle.dump(DTClassifier, open(filename, 'wb'))
```

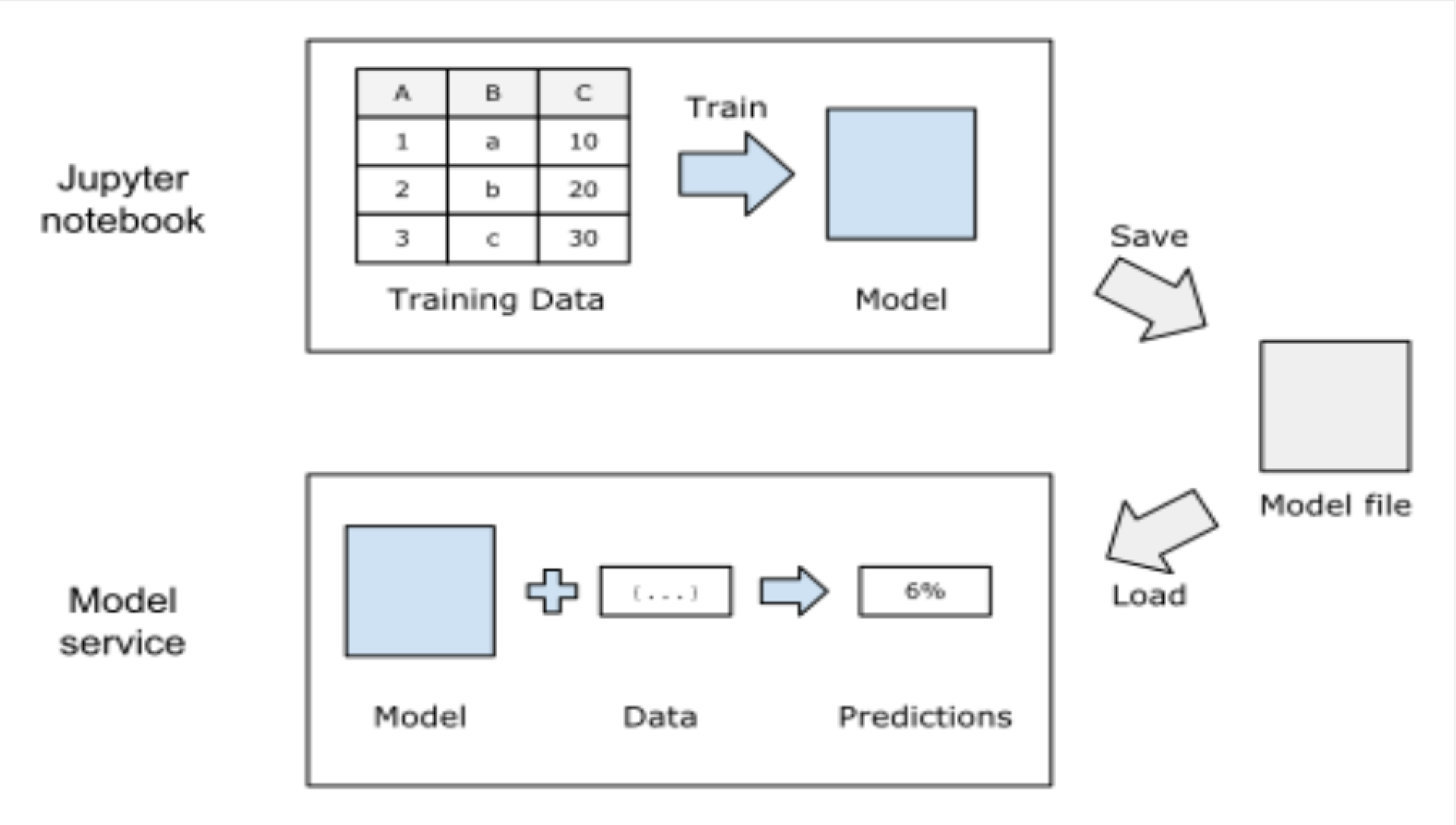
Operationalize

Here from the above graph, it is clearly visible that Glucose as a feature is the most important in this dataset.

- ❑ Saving the model

```
filename = 'finalized_model.pkl'
pickle.dump(DTClassifier, open(filename, 'wb'))
```

Operationalize



Operationalize

Technologies Used in this Project

Flask
web development,
one drop at a time

HEROKU

git --fast-version-control

Operationalize

.git	24/04/2022 17:21	File folder	
image	24/04/2022 13:30	File folder	
static	24/04/2022 17:14	File folder	
templates	24/04/2022 13:30	File folder	
app	24/04/2022 13:14	JetBrains PyCharm C...	2 KB
DiabeteClassification	23/04/2022 21:17	IPYNB File	238 KB
finalized_model.pkl	23/04/2022 21:17	PKL File	16 KB
Procfile	24/04/2022 12:54	File	1 KB
README	24/04/2022 13:30	MD File	1 KB
requirements	24/04/2022 12:54	Text Document	1 KB

Operationalize

```
$ git push -f heroku
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.32 MiB | 556.00 KiB/s, done.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the la
st build: python-3.10.4
remote:        To use a different version, see: https://devcenter.heroku.com/art
icles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.10.4
remote: -----> Installing pip 22.0.4, setuptools 60.10.0 and wheel 0.37.1
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 191.3M
remote: -----> Launching...
remote:        Released v4
remote:        https://diabetepredicto.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
```

Operationalize

Operationalize

Prediction: Wow ! You DON'T have diabetes.

Operationalize

You can play with the web app by follow this link

https://diabetepredicto.herokuapp.com/

The GitHub repository is attached below

https://github.com/Dilane-Kamga/DiabetePredictionApp.git

MERCI

Pour votre attention