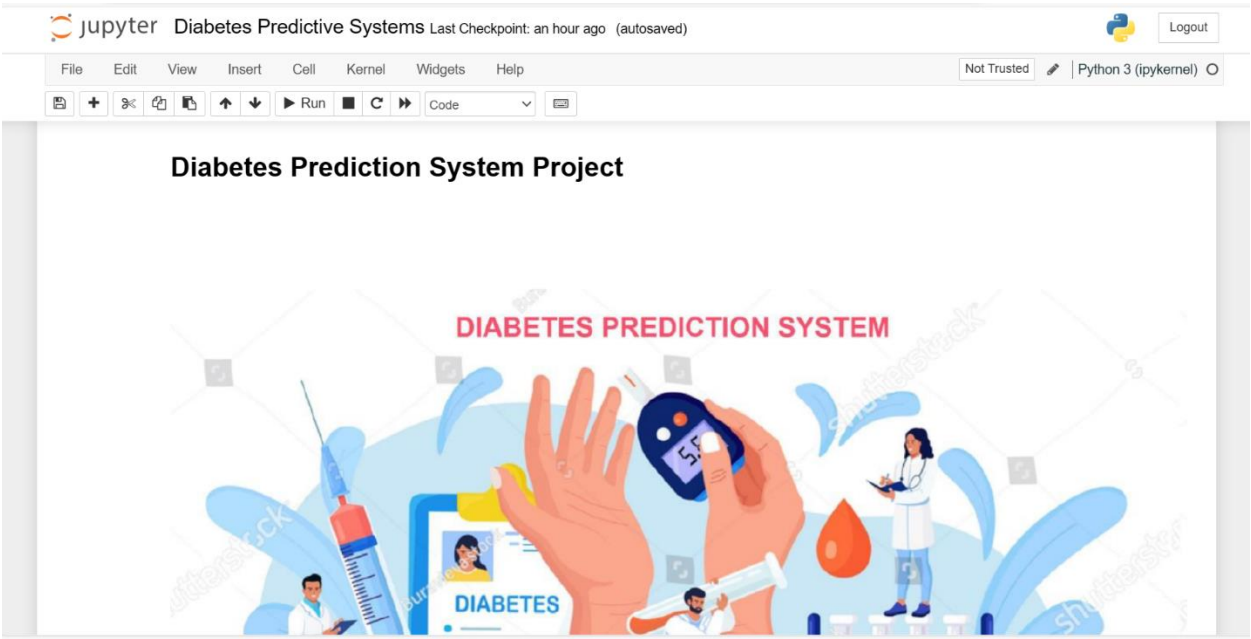
Project: Diabetes Diagnosis Prediction Model



Objective: The objective of this project is to create a machine learning model that will diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset or not.

Aims:

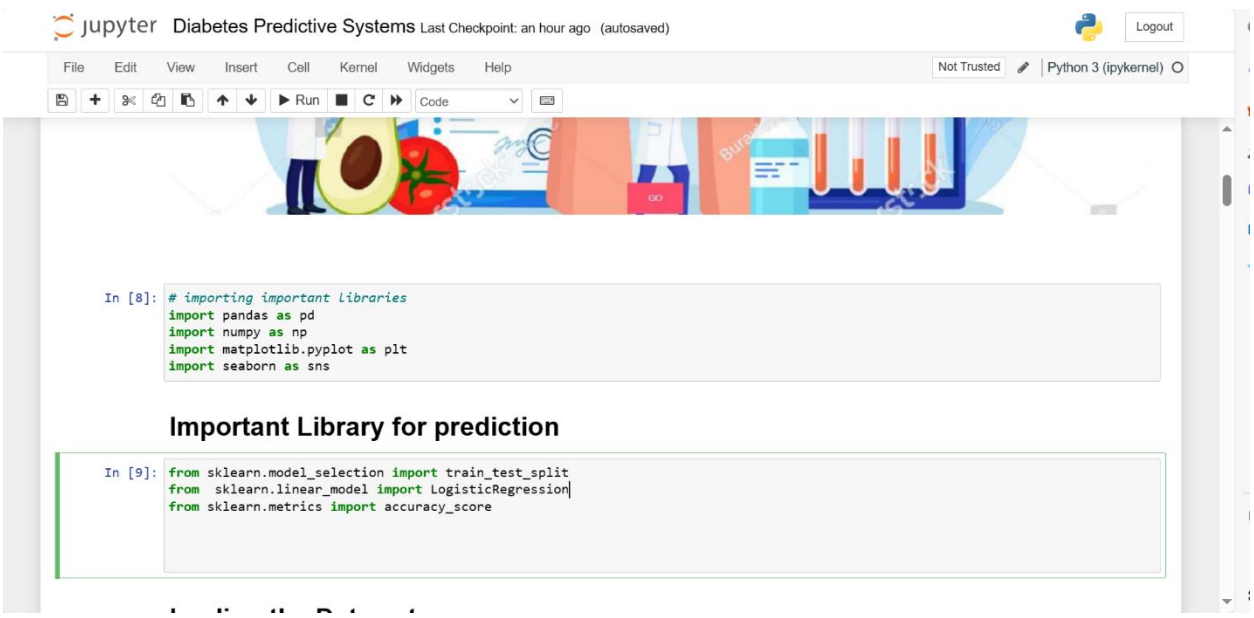
- Data Collection and Data Cleaning.
- Data Preprocessing and Visualization.
- Exploratory Data Analysis (EDA)
- Compare, Test, and implement the Machine Learning Algorithms
- Recommendations and future work.

Data Source: The National Institute of Diabetes and Digestive and Kidney Diseases. It is The Pima Women India dataset contains medical data of 768 women with 9 attributes to take note of and they are all at least 21 years of age.

Tools: Python
Model: Logistic Regression and Random forest algorithm

The target variable in the dataset is Outcome. Where 1 confirms the patient is diabetic and 0 shows negative results

I imported libraries that will be used to work with data from cleaning, visualization, and model-building



After library importing I imported the dataset, the dataset has to be looked at using the head and tail functions of five rows each.

jupyter Diabetes Predictive Systems Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

loading the Data set

In [11]:
```
Hombs=pd.read_csv("C:/Users/DELL/Desktop/Data Analytics/diabetes.csv")
```

Looking at the structure of the dataset using Head & Tail of 5 rows

In [12]:
```
Hombs.head(5)
```

Out[12]:

	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

In [12]:
```
Hombs.tail(5)
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0

The .shape function is to see the dimension of the data whereas this dataset contains 768 rows and 9 columns. I then verified the data by checking the null values using the .isnull method, false confirms the are no null values, and true confirms the null values hence in this dataset it seems there are no null values hence this is the perfect dataset to deal with for the project.

jupyter Diabetes Predictive Systems Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [31]:
```
print(Hombs.shape)
```

```
(768, 9)
```

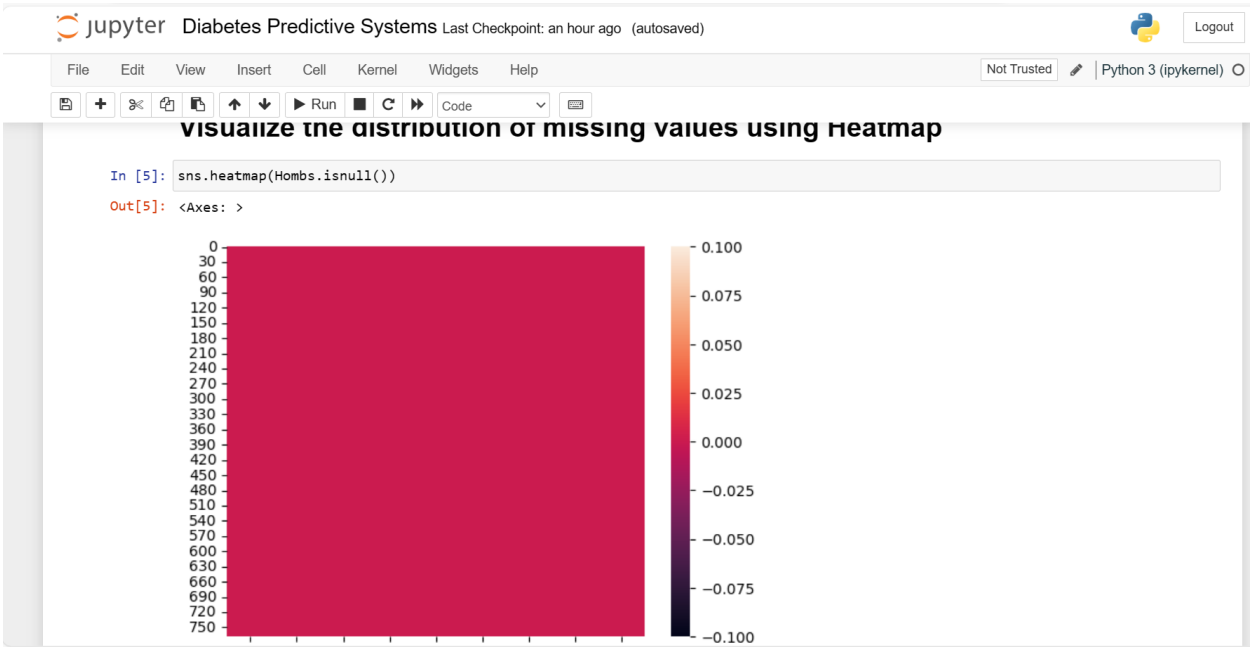
Checking the missing Values

In [30]:
```
Hombs.isnull() # checking if there is null value in the dataset-True means there is null value and false means no null values
```

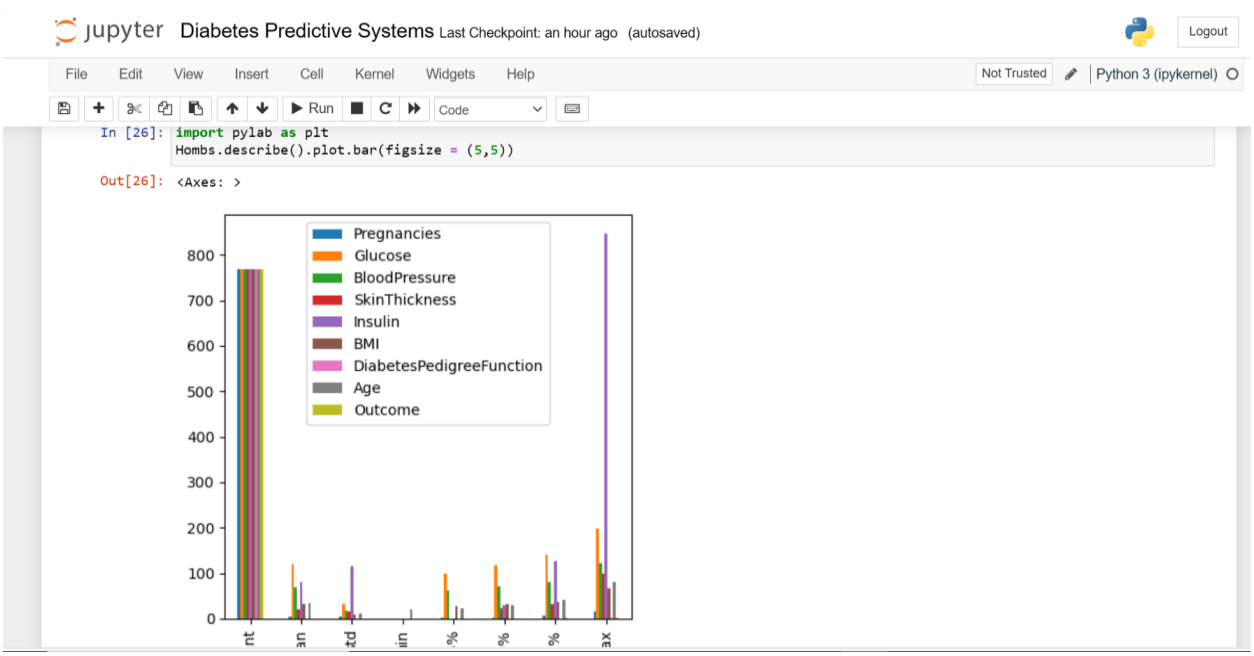
Out[30]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
763	False	False	False	False	False	False	False	False	False
764	False	False	False	False	False	False	False	False	False
765	False	False	False	False	False	False	False	False	False
766	False	False	False	False	False	False	False	False	False
767	False	False	False	False	False	False	False	False	False

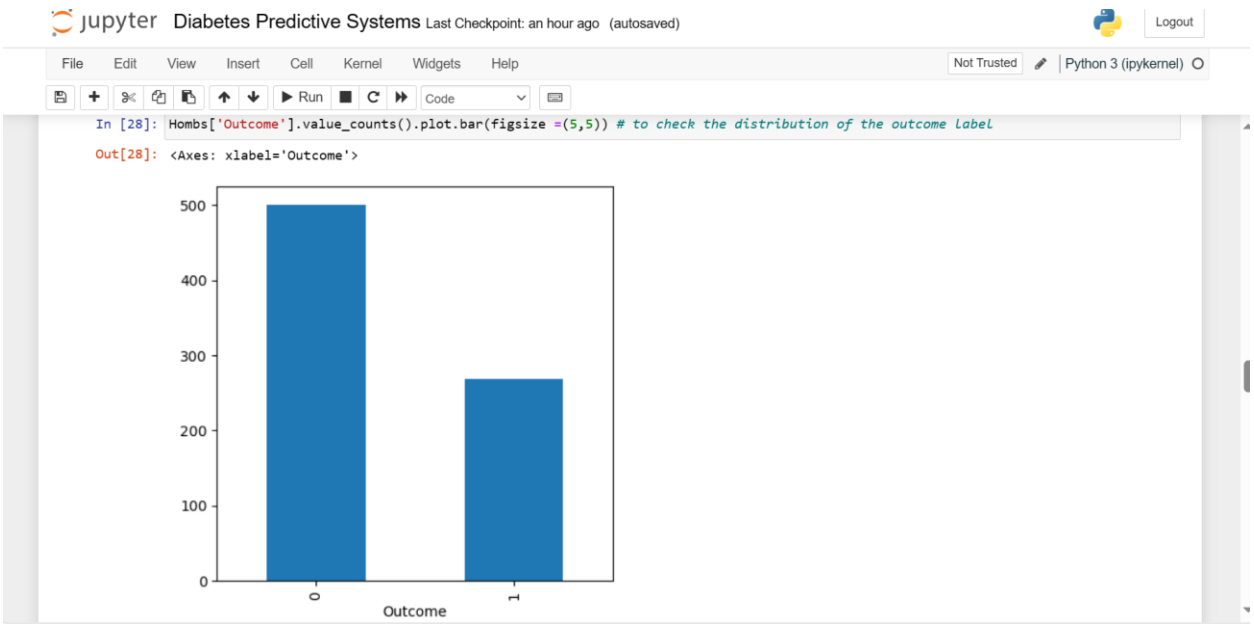
After checking the missing values I then check using a heatmap and the data looks perfectly



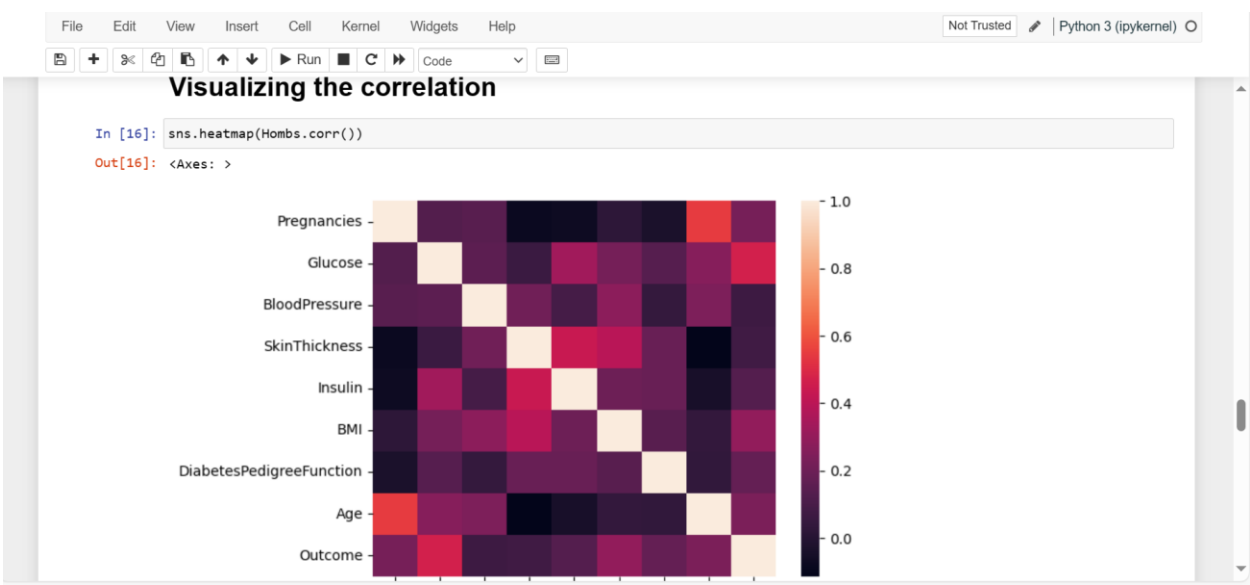
I also looked at the statistical distribution of the variables meaning checking on the mean, mode,%ntile distribution of the data.



Outcome is the measure of diabetes on this data hence I also checked on the distribution of positives or negatives. This dataset it shows that the number of negative outcomes is higher than positive



The visual correlation show that glucose is highly correlated in this dataset



After exploring and visualizing the data, it's time to build a model, hence I will start with logistic regression followed by random forest. I separated the dataset on the outcome column because it's the one with the variable to predict.

I will split the data into 80% training and 20% testing. The higher the training the accuracy of the model

jupyter Diabetes Predictive Systems Last Checkpoint: 2 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Now Lets build a model

In [13]:
```
X=Hombs.drop('Outcome',axis=1)  # we are droping the OUTCOME column because is the one with the classification labels
Y=Hombs['Outcome']              # 0 means there is no diabetes detected and 1 means diabetes detected
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2)  # 80% training data and 20% testing data
```

Train the model Using Logistic regression

In [14]:
```
model = LogisticRegression(solver='lbfgs', max_iter=1000) # solver='lbfgs', max_iter=1000- this statement helps to avoid converge
model.fit(X_train,Y_train)
```

Out[14]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```

Making Prediction

In [15]:
```
prediction=model.predict(X_test)
```

In [16]:
```
print(prediction)
```

jupyter Diabetes Predictive Systems Last Checkpoint: 2 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Making Prediction

In [15]:
```
prediction=model.predict(X_test)
```

In [16]:
```
print(prediction)
```

```
[0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0
 0 0 1 1 0 1 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 0 0 1 0 0 0 0
 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 0 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0
 0 0 0 0 0 1]
```

In [17]:
```
accuracy=accuracy_score(prediction,Y_test)
```

In [18]:
```
print(accuracy)
```

```
0.7532467532467533
```

Train the model using random forest

In [32]:
```
# Fitting Random Forest Regression to the dataset
# import the regressor
from sklearn.ensemble import RandomForestRegressor
```

jupyter Diabetes Predictive Systems Last Checkpoint: 2 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Train the model using random forest

In [32]:
```
# Fitting Random Forest Regression to the dataset
# import the regressor
from sklearn.ensemble import RandomForestRegressor

# create regressor object
regressor = RandomForestRegressor(n_estimators=100,
                                  random_state=0)

# fit the regressor with x and y data
regressor.fit(X_train,Y_train)
```

Out[32]:
```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

In [39]:
```
HombsModel=RandomForestRegressor(random_state=0)
```

In [40]:
```
HombsModel.fit(X_train,Y_train)
```

Out[40]:
```
RandomForestRegressor
RandomForestRegressor(random_state=0)
```

jupyter Diabetes Predictive Systems Last Checkpoint: 2 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Predicting the model using Random forest

In [43]:
```
predict=HombsModel.predict(X_test)
```

In [44]:
```
print(predict)
```

```
[0.48 0.12 0.04 0.57 0.59 0.8  0.   0.16 0.93 0.47 0.05 0.45 0.02 0.26
 0.88 0.9  0.1  0.61 0.02 0.49 0.2  0.01 0.57 0.34 0.78 0.14 0.04 0.7
 0.02 0.   0.92 0.3  0.08 0.22 0.16 0.05 0.37 0.48 0.35 0.42 0.56 0.5
 0.59 0.18 0.73 0.52 0.71 0.   0.27 0.07 0.67 0.35 0.48 0.02 0.56 0.93
 0.63 0.87 0.88 0.05 0.81 0.9  0.69 0.69 0.04 0.72 0.47 0.15 0.52 0.88
 0.24 0.06 0.47 0.42 0.29 0.91 0.42 0.24 0.01 0.48 0.03 0.29 0.56 0.96
 0.17 0.04 0.03 0.33 0.13 0.1  0.63 0.68 0.63 0.43 0.69 0.61 0.26 0.71
 0.77 0.33 0.47 0.75 0.63 0.94 0.03 0.26 0.11 0.43 0.66 0.28 0.   0.
 0.   0.05 0.12 0.   0.02 0.06 0.63 0.22 0.68 0.09 0.52 0.71 0.55 0.04
 0.11 0.73 0.36 0.83 0.36 0.61 0.16 0.   0.82 0.63 0.05 0.01 0.74 0.64
 0.12 0.03 0.12 0.09 0.07 0.5  0.08 0.25 0.72 0.12 0.06 0.1  0.35 0.75]
```

In [ ]:

The prediction score is 75.3% and the score for the random forest is 75% hence I would suggest going for the Logistic regression algorithm because it's easy to understand and fast to process.

Work to be done:

I will install Django through Pycharm to create an interface for the application that I will link with the model in order to allow users to predict the outcome using a mobile or web application.