Minor Project

Verzeo ML May Batch-1

Diabetes Prediction

By,
Prince P. Pipariya
princeppipariya24@gmail.com

Problem Statement :

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

Packages:

- pandas
- numpy
- sklearn
- matplotlib

IDE:

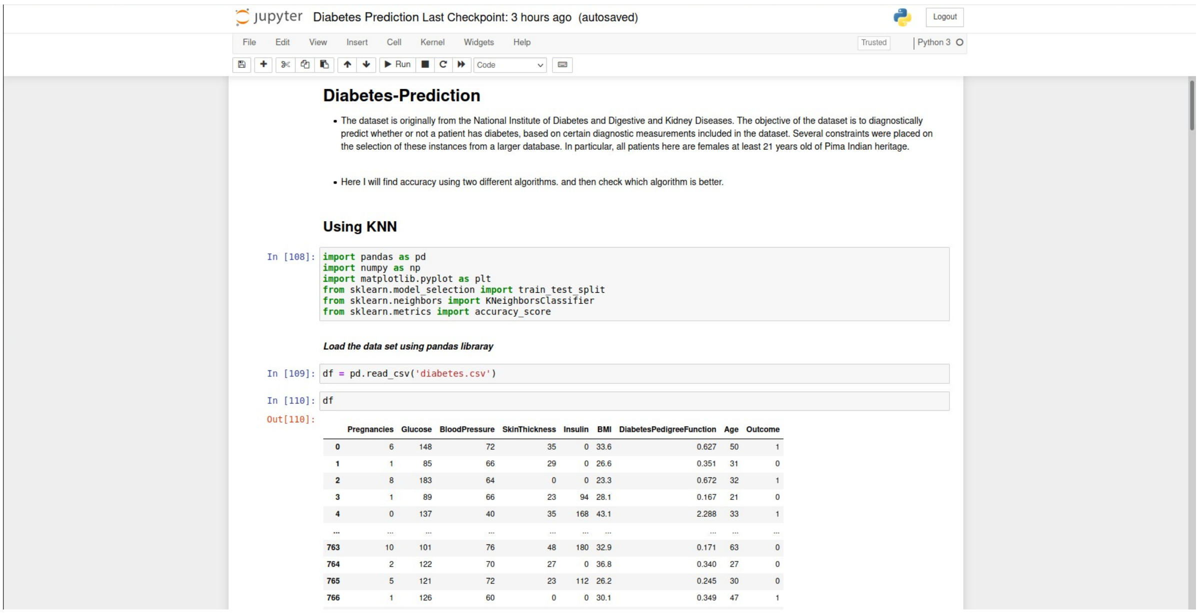
- Jupyter Notebook
- Google Colab (Just for Testing Purpose)

Details:

In this project, we required to find that any lady whose age is more than 21 has diabetes or not so for this prediction we must need of data set and it also provided. Here for this problem statement we have to apply classification techanique and for that i have tried two different classification like knn and logistic regression and find out which technique is better.

Working & Screen shots:

- First i applied knn algorithm.



Diabetes-Prediction

- The dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

- Here I will find accuracy using two different algorithms. and then check which algorithm is better.

Using KNN

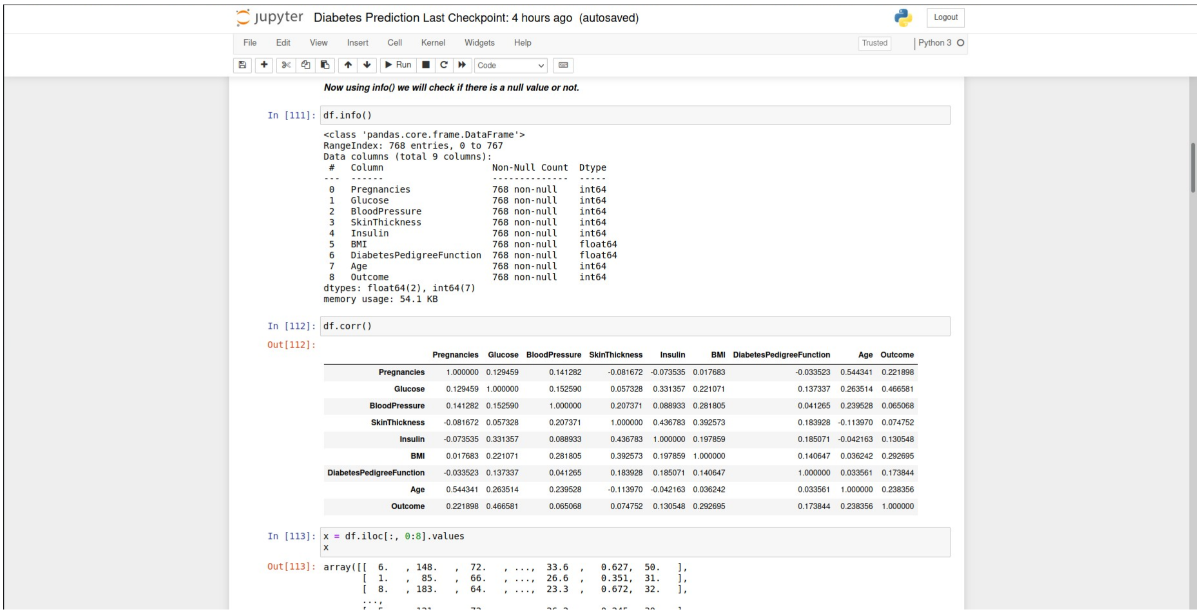
```
In [108]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          from sklearn.model_selection import train_test_split
          from sklearn.neighbors import KNeighborsClassifier
          from sklearn.metrics import accuracy_score
```

Load the data set using pandas libraray

```
In [109]: df = pd.read_csv('diabetes.csv')

In [110]: df
Out[110]:
```

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1

- Now i go for that there is null value or not.



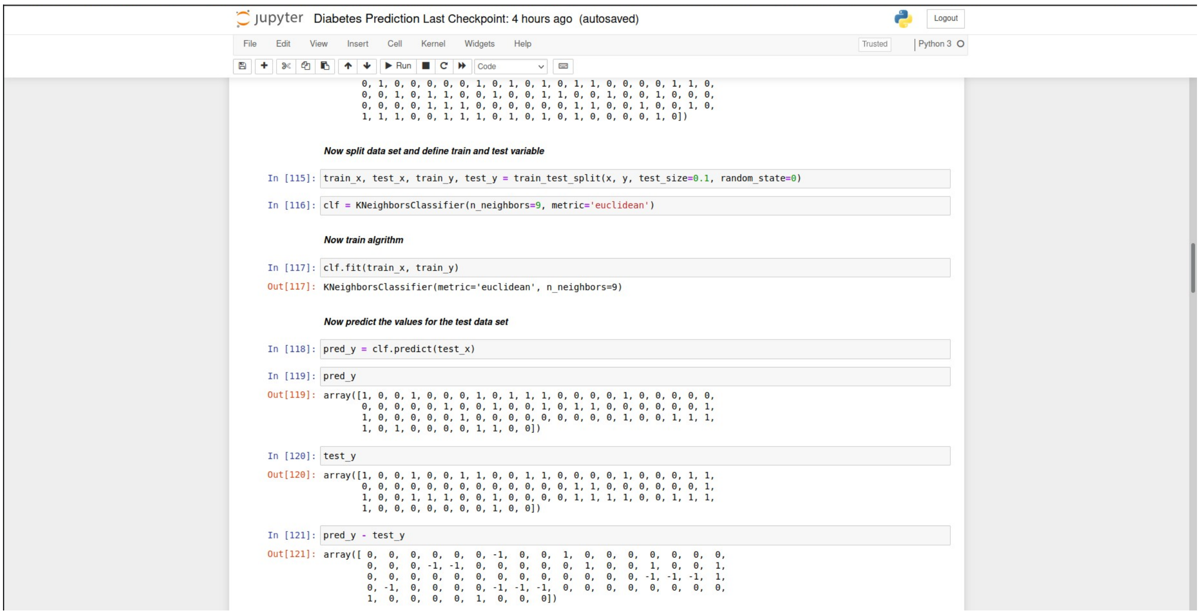
Now using info() we will check if there is a null value or not.

```
In [111]: 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

In [112]: df.corr()
Out[112]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

```
In [113]: x = df.iloc[:, 0:8].values
          x

Out[113]: array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
                 [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
                 [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
                 ...,
```

- Now split data and apply knn algrithm for k=9 and find accuracy.



```
          0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
          0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
          0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
          1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

Now split data set and define train and test variable

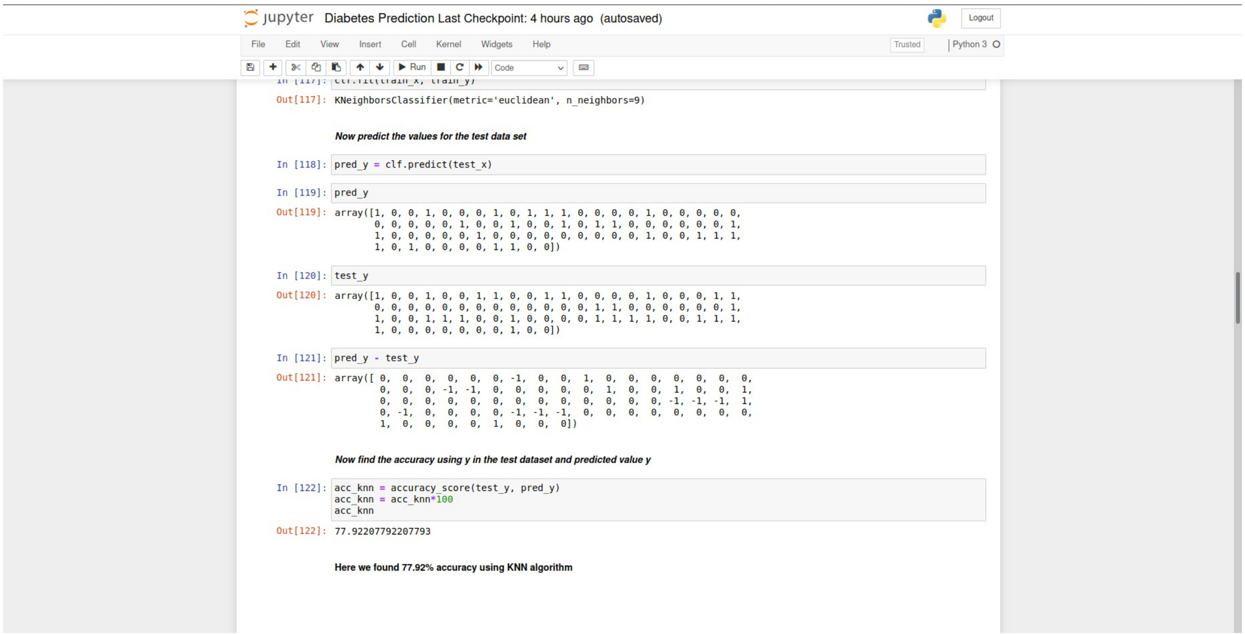
```
In [115]: train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.1, random_state=0)

In [116]: clf = KNeighborsClassifier(n_neighbors=9, metric='euclidean')
```

Now train algrithm

```
In [117]: clf.fit(train_x, train_y)

Out[117]: KNeighborsClassifier(metric='euclidean', n_neighbors=9)
```

Now predict the values for the test data set

```
In [118]: pred_y = clf.predict(test_x)

In [119]: pred_y

Out[119]: array([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
                 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0])

In [120]: test_y

Out[120]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])

In [121]: pred_y - test_y

Out[121]: array([ 0,  0,  0,  0,  0,  0, -1,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0, -1, -1,  0,  0,  0,  0,  0,  1,  0,  0,  1,  0,  0,  1,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  1,
                  0, -1,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,
                  1,  0,  0,  0,  0,  1,  0,  0,  0])
```



```
In [117]: clf.fit(train_x, train_y)

Out[117]: KNeighborsClassifier(metric='euclidean', n_neighbors=9)
```

Now predict the values for the test data set

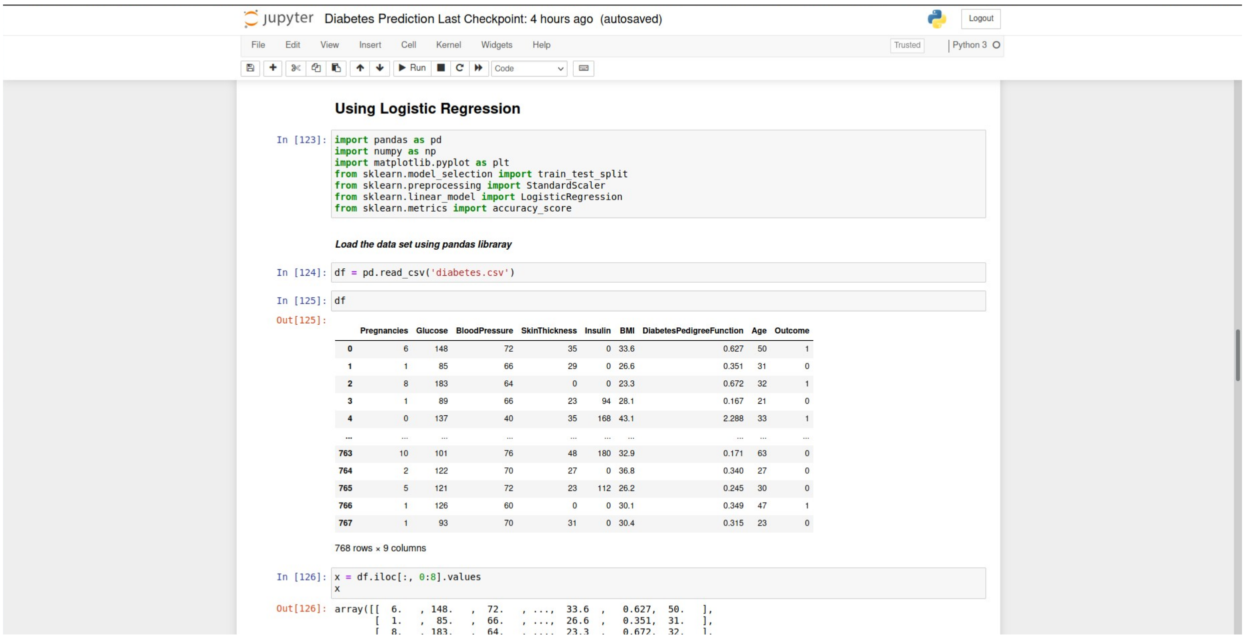
```
In [118]: pred_y = clf.predict(test_x)

In [119]: pred_y

Out[119]: array([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1,
                 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0])

In [120]: test_y

Out[120]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])

In [121]: pred_y - test_y

Out[121]: array([ 0,  0,  0,  0,  0,  0, -1,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0, -1, -1,  0,  0,  0,  0,  0,  1,  0,  0,  1,  0,  0,  1,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  1,
                  0, -1,  0,  0,  0,  0, -1, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,
                  1,  0,  0,  0,  0,  1,  0,  0,  0])
```

Now find the accuracy using y in the test dataset and predicted value y

```
In [122]: acc_knn = accuracy_score(test_y, pred_y)
          acc_knn = acc_knn*100
          acc_knn

Out[122]: 77.92207792207793
```

Here we found 77.92% accuracy using KNN algorithm

- Second method is Logistic regression.



Using Logistic Regression

```
In [123]: import pandas as pd
          import numpy as np
          import matplotlib.pyplot as plt
          from sklearn.model_selection import train_test_split
          from sklearn.preprocessing import StandardScaler
          from sklearn.linear_model import LogisticRegression
          from sklearn.metrics import accuracy_score
```

Load the data set using pandas libraray

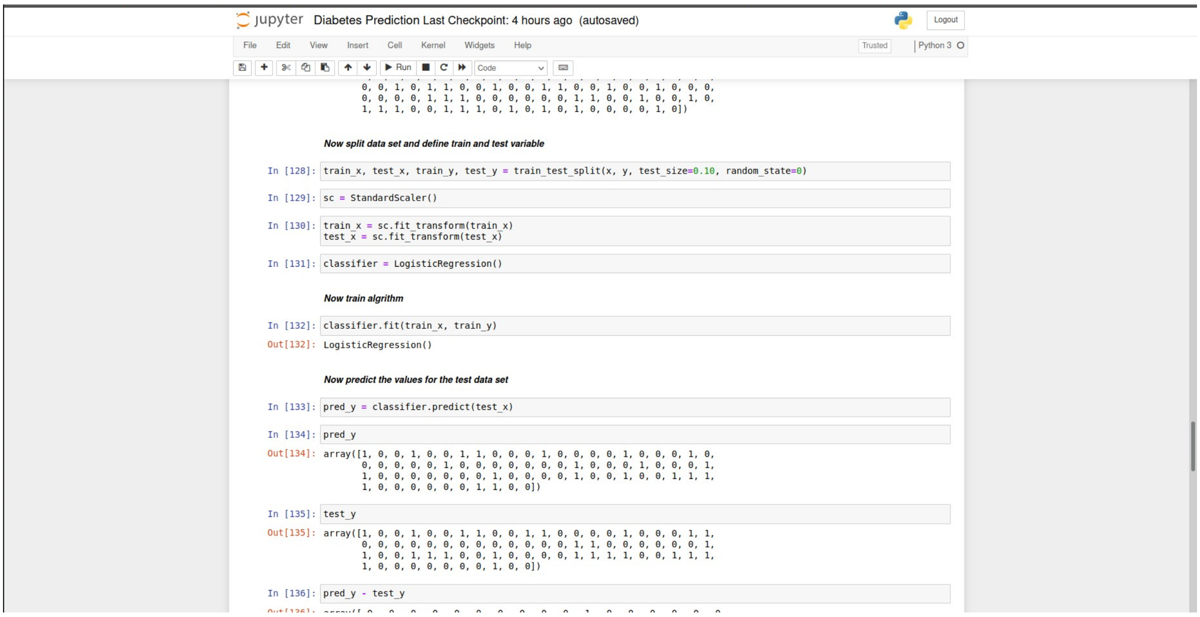
```
In [124]: df = pd.read_csv('diabetes.csv')

In [125]: df
Out[125]:
```

	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
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

```
In [126]: x = df.iloc[:, 0:8].values
          x
Out[126]: array([[  6.   , 148.   ,  72.   , ...,  33.6  ,   0.627,  50.   ],
                 [  1.   ,  85.   ,  66.   , ...,  26.6  ,   0.351,  31.   ],
                 [  8.   , 183.   ,  64.   , ...,  23.3  ,   0.672,  32.   ],
```

- Now split data and apply logistic algorithm and find accuracy.



```
                 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
                 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
                 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

Now split data set and define train and test variable

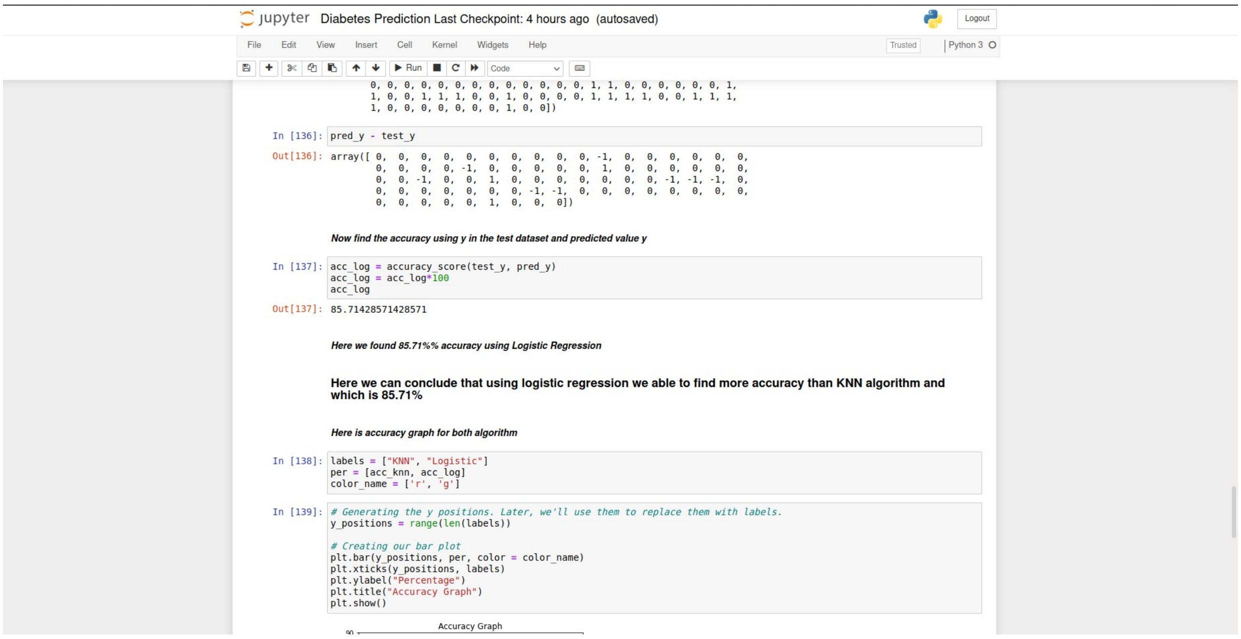
```
In [128]: train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.10, random_state=0)

In [129]: sc = StandardScaler()

In [130]: train_x = sc.fit_transform(train_x)
          test_x = sc.fit_transform(test_x)

In [131]: classifier = LogisticRegression()
```

Now train algrithm

```
In [132]: classifier.fit(train_x, train_y)
Out[132]: LogisticRegression()
```

Now predict the values for the test data set

```
In [133]: pred_y = classifier.predict(test_x)

In [134]: pred_y
Out[134]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
                 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,
                 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0])

In [135]: test_y
Out[135]: array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1,
                 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])

In [136]: pred_y - test_y
Out[136]: array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,
```



```
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
                1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1,
                1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0])
```

```
In [136]: pred_y - test_y
```

```
Out[136]: array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,
                  0,  0, -1,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0, -1, -1, -1,  0,
                  0,  0,  0,  0,  0,  0,  0, -1, -1,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  1,  0,  0,  0])
```

Now find the accuracy using y in the test dataset and predicted value y

```
In [137]: acc_log = accuracy_score(test_y, pred_y)
          acc_log = acc_log*100
          acc_log
```

```
Out[137]: 85.71428571428571
```

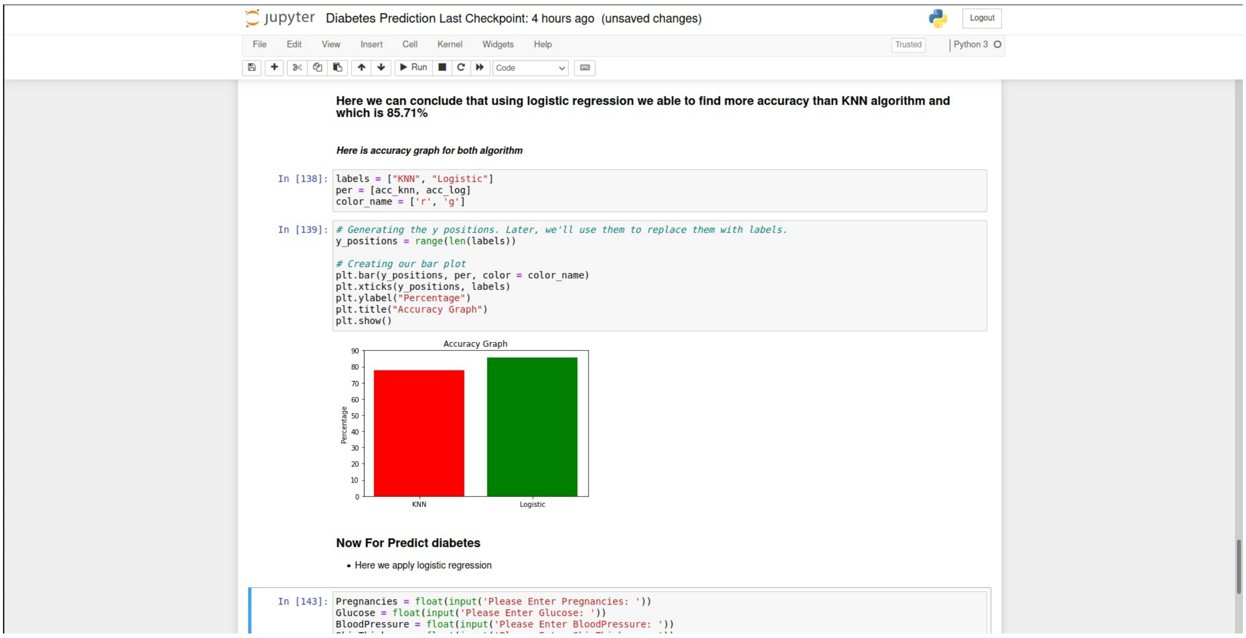
Here we found 85.71%% accuracy using Logistic Regression

Here we can conclude that using logistic regression we able to find more accuracy than KNN algorithm and which is 85.71%

Here is accuracy graph for both algorithm

```
In [138]: labels = ["KNN", "Logistic"]
          per = [acc_knn, acc_log]
          color_name = ['r', 'g']
```

```
In [139]: # Generating the y positions. Later, we'll use them to replace them with labels.
          y_positions = range(len(labels))

          # Creating our bar plot
          plt.bar(y_positions, per, color = color_name)
          plt.xticks(y_positions, labels)
          plt.ylabel("Percentage")
          plt.title("Accuracy Graph")
          plt.show()
```



Here we can conclude that using logistic regression we able to find more accuracy than KNN algorithm and which is 85.71%

Here is accuracy graph for both algorithm

```
In [138]: labels = ["KNN", "Logistic"]
          per = [acc_knn, acc_log]
          color_name = ['r', 'g']
```

```
In [139]: # Generating the y positions. Later, we'll use them to replace them with labels.
          y_positions = range(len(labels))

          # Creating our bar plot
          plt.bar(y_positions, per, color = color_name)
          plt.xticks(y_positions, labels)
          plt.ylabel("Percentage")
          plt.title("Accuracy Graph")
          plt.show()
```

Now For Predict diabetes

- Here we apply logistic regression

```
In [143]: Pregnancies = float(input('Please Enter Pregnancies: '))
          Glucose = float(input('Please Enter Glucose: '))
          BloodPressure = float(input('Please Enter BloodPressure: '))
```

Conclusion:

Here we can conclude that using logistic regression we able to find more accuracy than KNN algorithm and which is 85.71%