Capstone - Healthcare

Project 2 :

Name : Keerthi Rajan

DESCRIPTION

Problem Statement

- NIDDK (National Institute of Diabetes and Digestive and Kidney Diseases) research creates knowledge about and treatments for the most chronic, costly, and consequential diseases.
- The dataset used in this project is originally from NIDDK. The objective is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Build a model to accurately predict whether the patients in the dataset have diabetes or not.

Project Task: Week 1

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

• Glucose

• BloodPressure

• SkinThickness

• Insulin

• BMI

2. Visually explore these variables using histograms. Treat the missing values accordingly.

3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

Project Task: Week 2

Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.

2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

3. Perform correlation analysis. Visually explore it using a heat map.

Project Task: Week 3

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.

2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

Project Task: Week 4

Data Modeling:

1. Create a classification report by analyzing sensitivity, specificity, AUC (ROC curve), etc. Please be descriptive to explain what values of these parameter you have used.

Data Reporting:

2. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

    a. Pie chart to describe the diabetic or non-diabetic population

    b. Scatter charts between relevant variables to analyze the relationships

    c. Histogram or frequency charts to analyze the distribution of the data

d. Heatmap of correlation analysis among the relevant variables

e. Create bins of these age values: 20-25, 25-30, 30-35, etc. Analyze different variables for these age brackets using a bubble chart.

Analysis

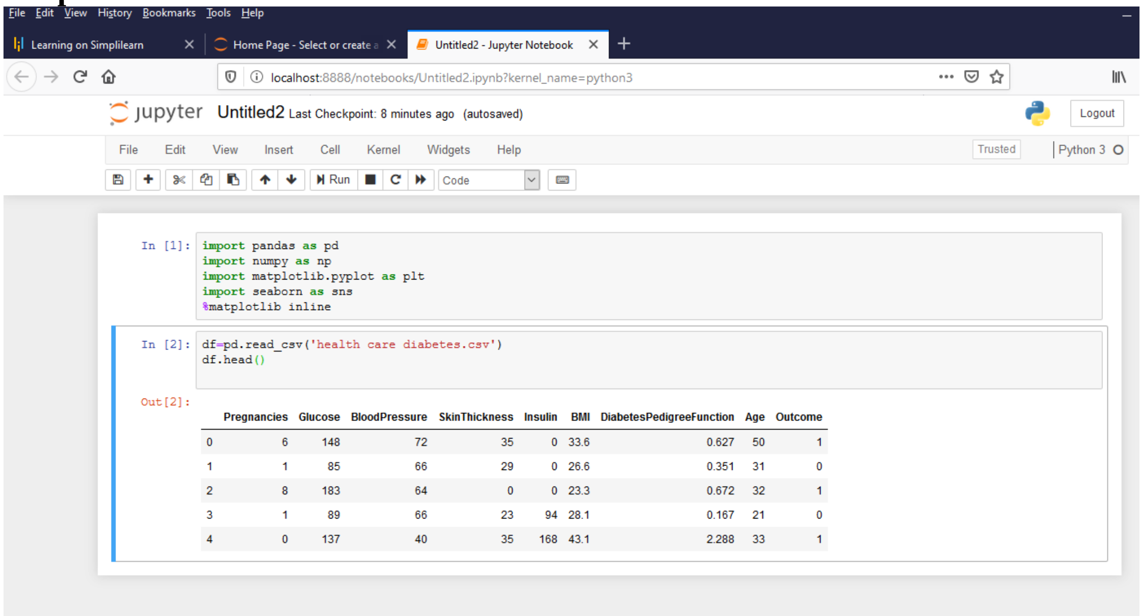
Project Task : Week 1

Code And Screenshots

Code

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlibinline


df=pd.read_csv('health care diabetes.csv')

df.head()
```

Output



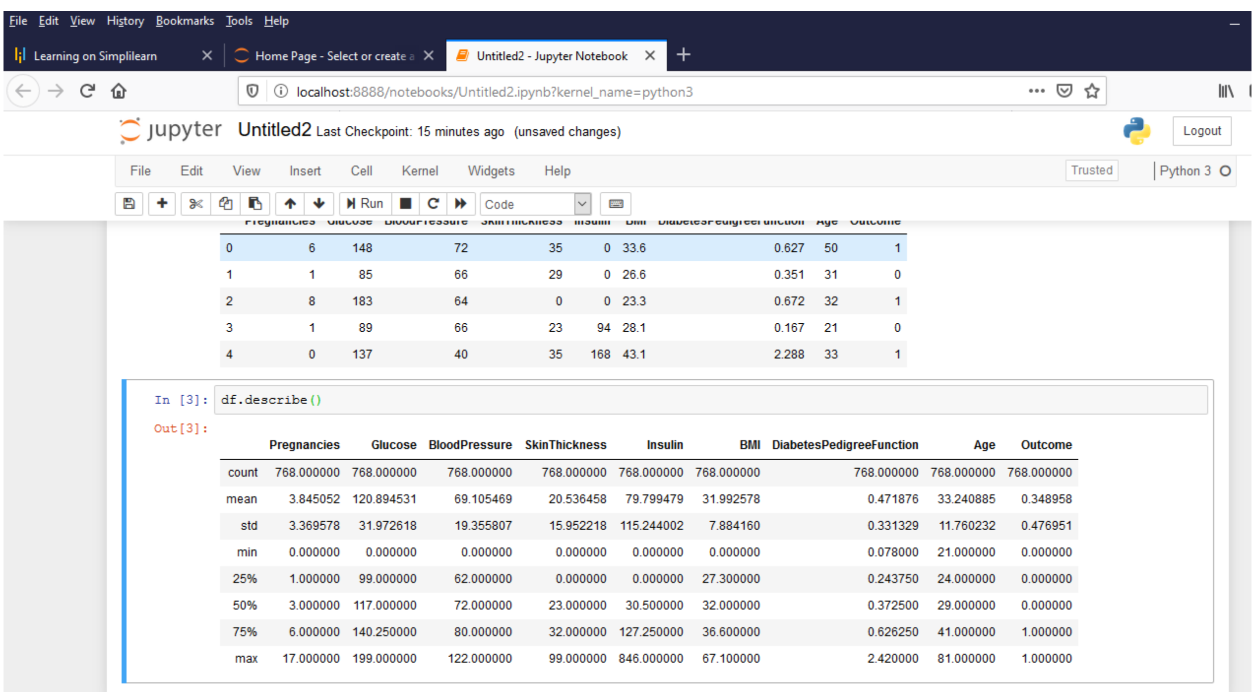
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

In [2]: df=pd.read_csv('health care diabetes.csv')
        df.head()
```

Out[2]:

	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

Code

df.describe()



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 [3]: df.describe()
```

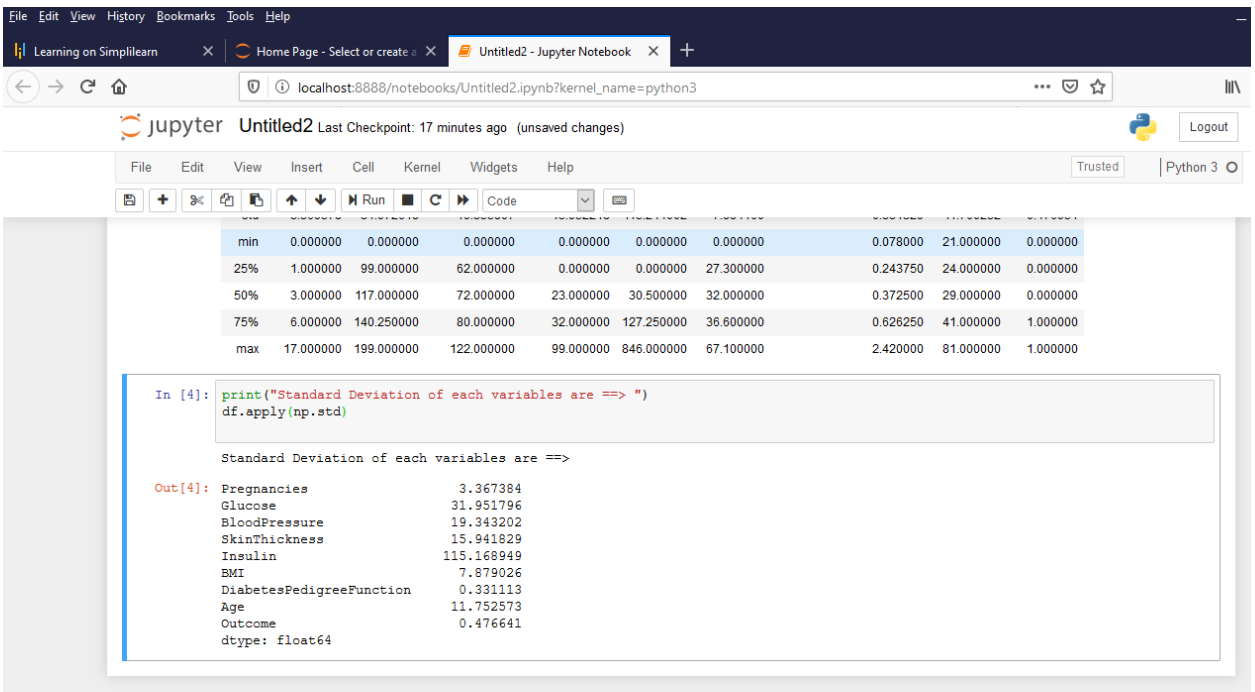
Out[3]:

	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

Code

```
print("Standard Deviation of each variables are ==> ")

df.apply(np.std)
```

Output



Code

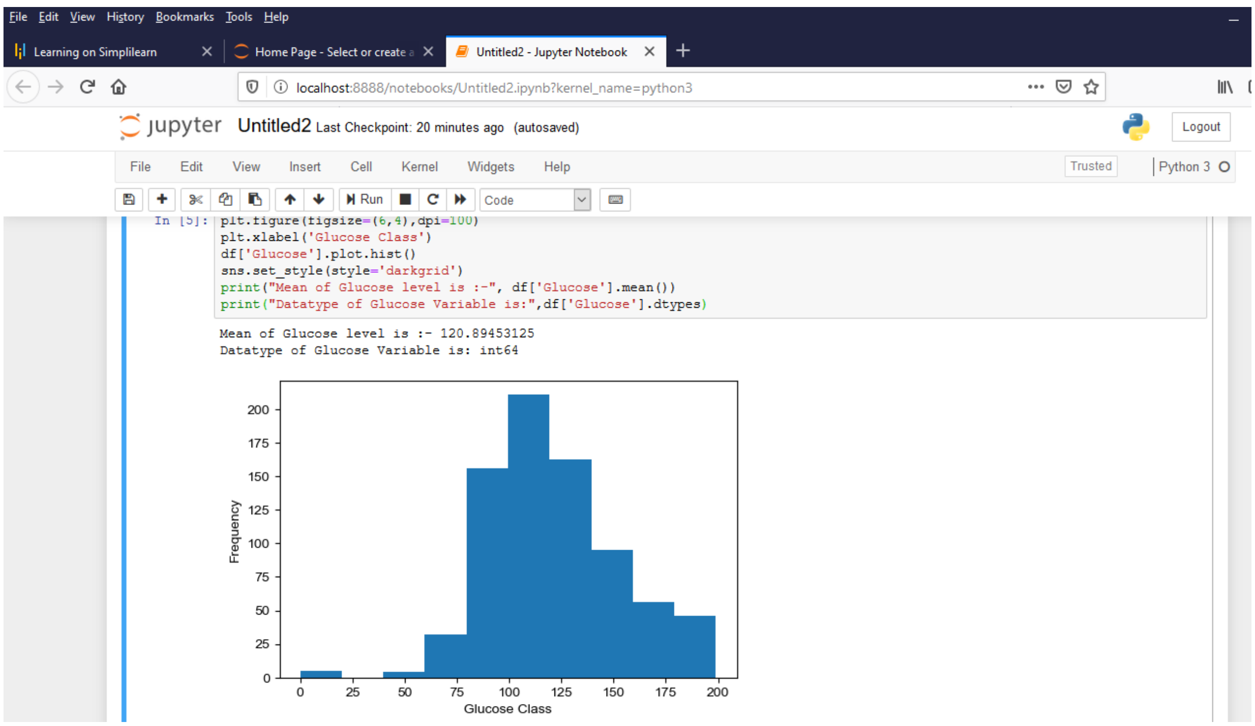
```
plt.figure(figsize=(6,4),dpi=100)

plt.xlabel('Glucose Class')

df['Glucose'].plot.hist()
```

sns.set_style(style='darkgrid')

print("Mean of Glucose level is :-", df['Glucose'].mean())

print("Datatype of Glucose Variable is:",df['Glucose'].dtypes)

Output



Code

df['Glucose']=df['Glucose'].replace(0,df['Glucose'].mean())

plt.figure(figsize=(6,4),dpi=100)

plt.xlabel('BloodPressure Class')

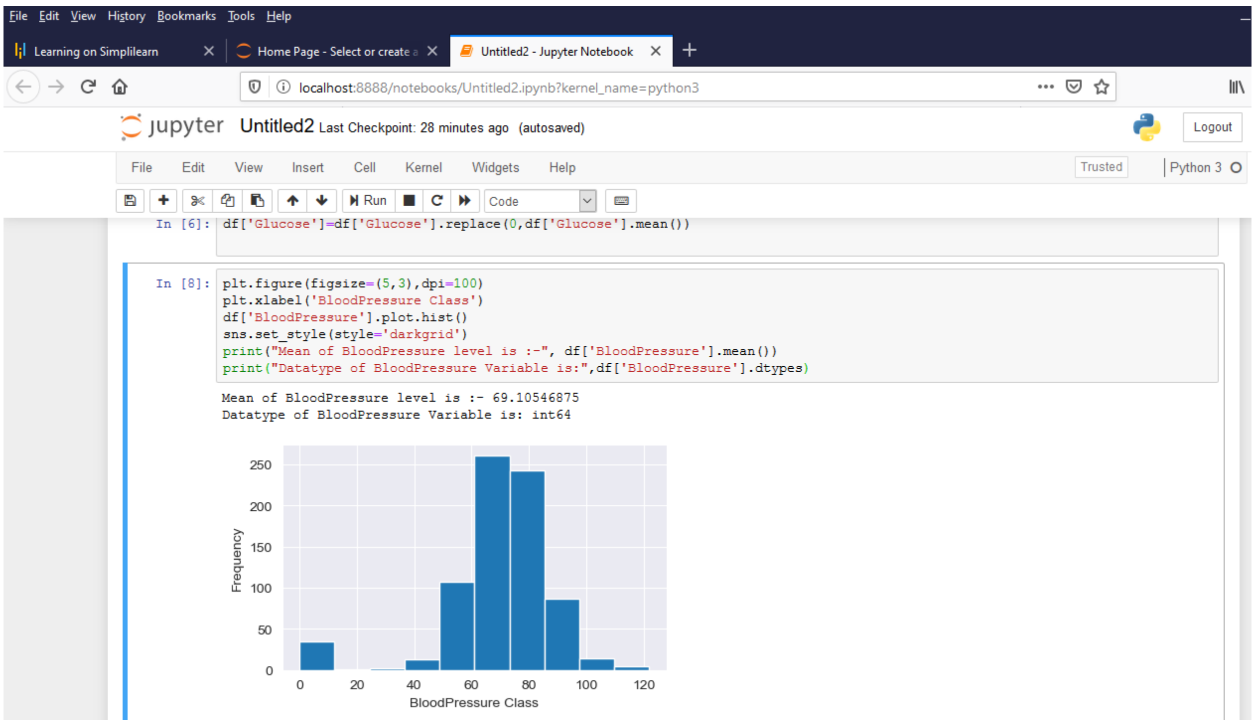
df['BloodPressure'].plot.hist()

sns.set_style(style='darkgrid')

print("Mean of BloodPressure level is :-", df['BloodPressure'].mean())

print("Datatype of BloodPressure Variable is:",df['BloodPressure'].dtypes)

Output



Code

df['BloodPressure']=df['BloodPressure'].replace(0,df['BloodPressure'].mean())

plt.figure(figsize=(5,3),dpi=100)

plt.xlabel('SkinThickness Class')

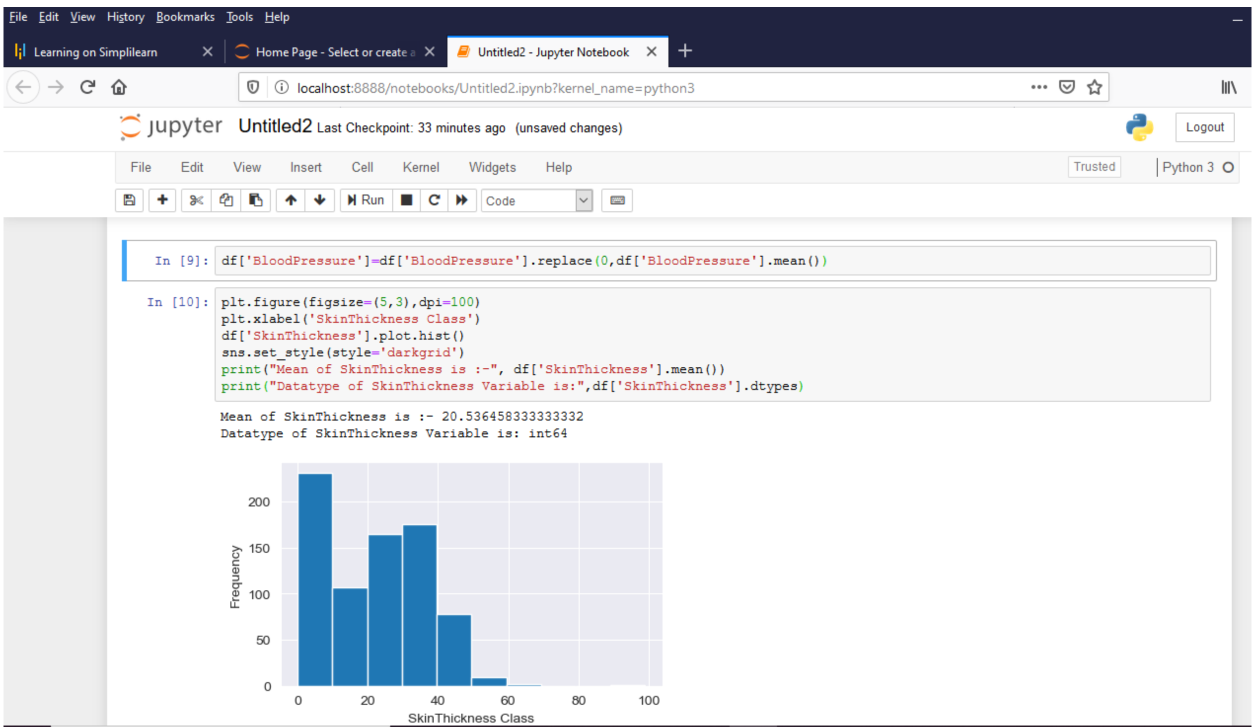
df['SkinThickness'].plot.hist()

sns.set_style(style='darkgrid')

print("Mean of SkinThickness is :-", df['SkinThickness'].mean())

print("Datatype of SkinThickness Variable is:",df['SkinThickness'].dtypes)

Output



Code

df['SkinThickness']=df['SkinThickness'].replace(0,df['SkinThickness'].mean())

plt.figure(figsize=(5,3),dpi=100)

plt.xlabel('Insulin Class')

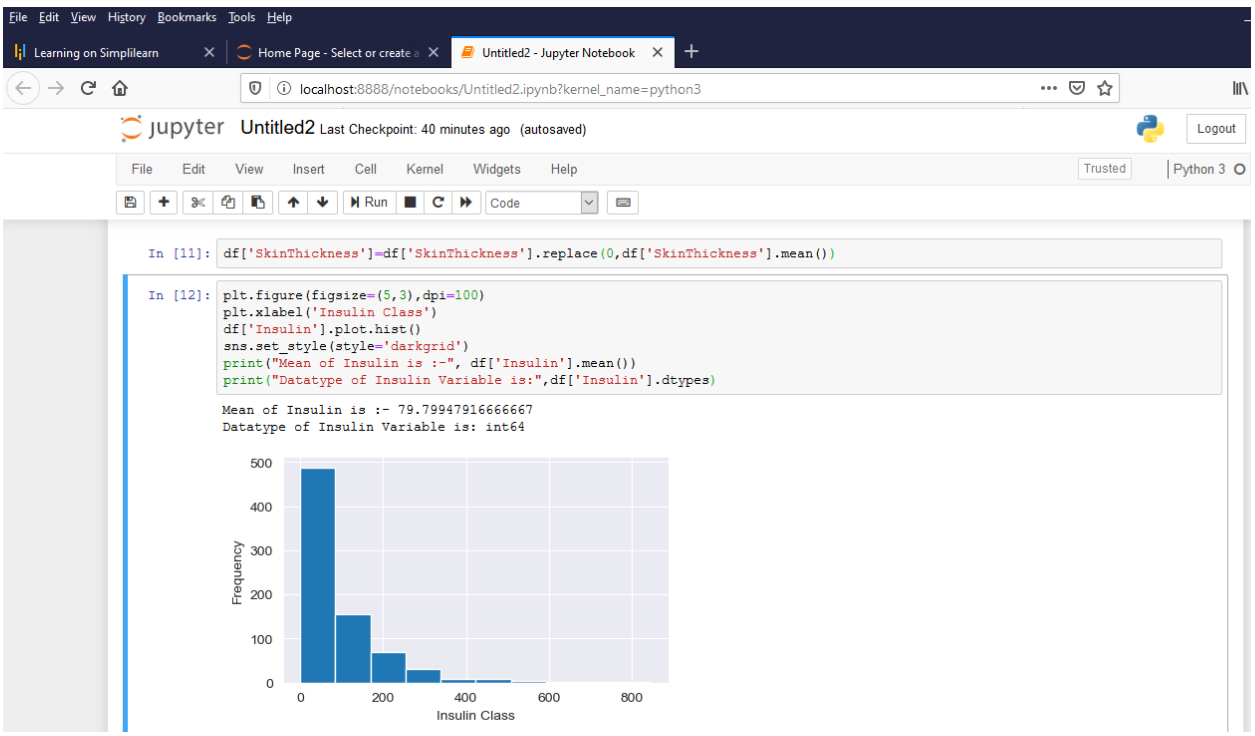
df['Insulin'].plot.hist()

sns.set_style(style='darkgrid')

print("Mean of Insulin is :-", df['Insulin'].mean())

print("Datatype of Insulin Variable is:",df['Insulin'].dtypes)

Output



Code

df['Insulin']=df['Insulin'].replace(0,df['Insulin'].mean())

plt.figure(figsize=(5,3),dpi=100)

plt.xlabel('BMI Class')

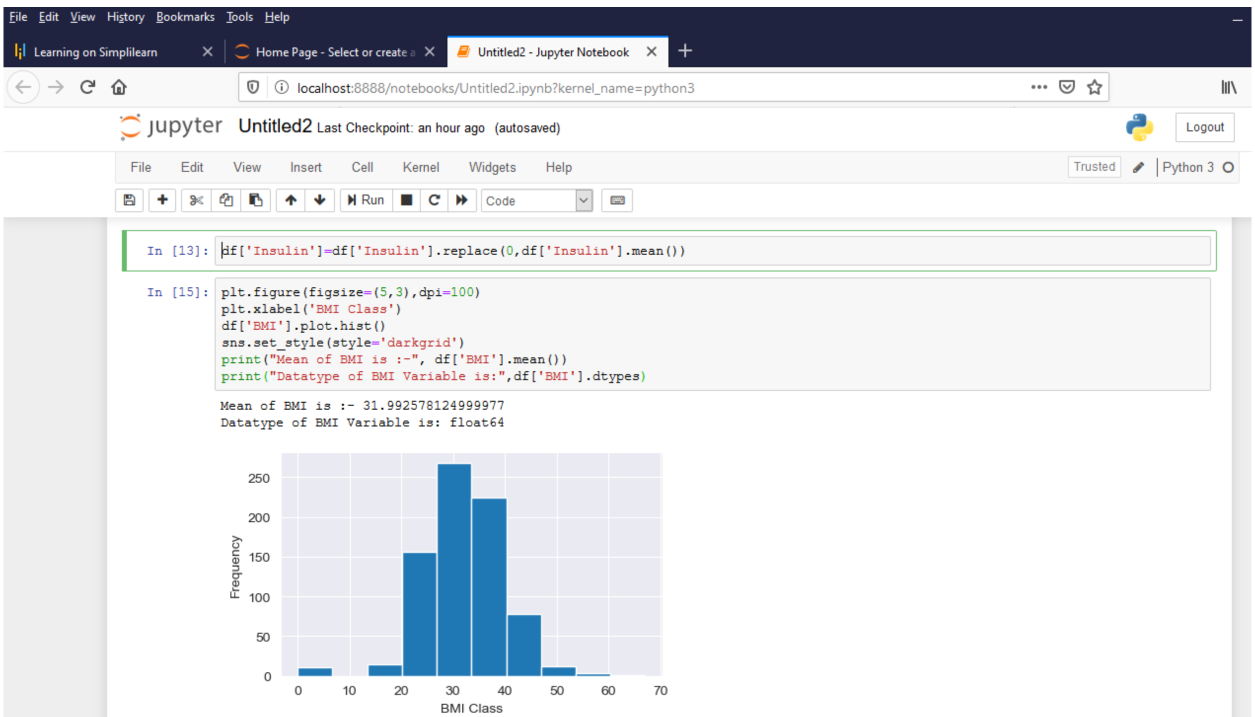
df['BMI'].plot.hist()

sns.set_style(style='darkgrid')

print("Mean of BMI is :-", df['BMI'].mean())

print("Datatype of BMI Variable is:",df['BMI'].dtypes)

Output



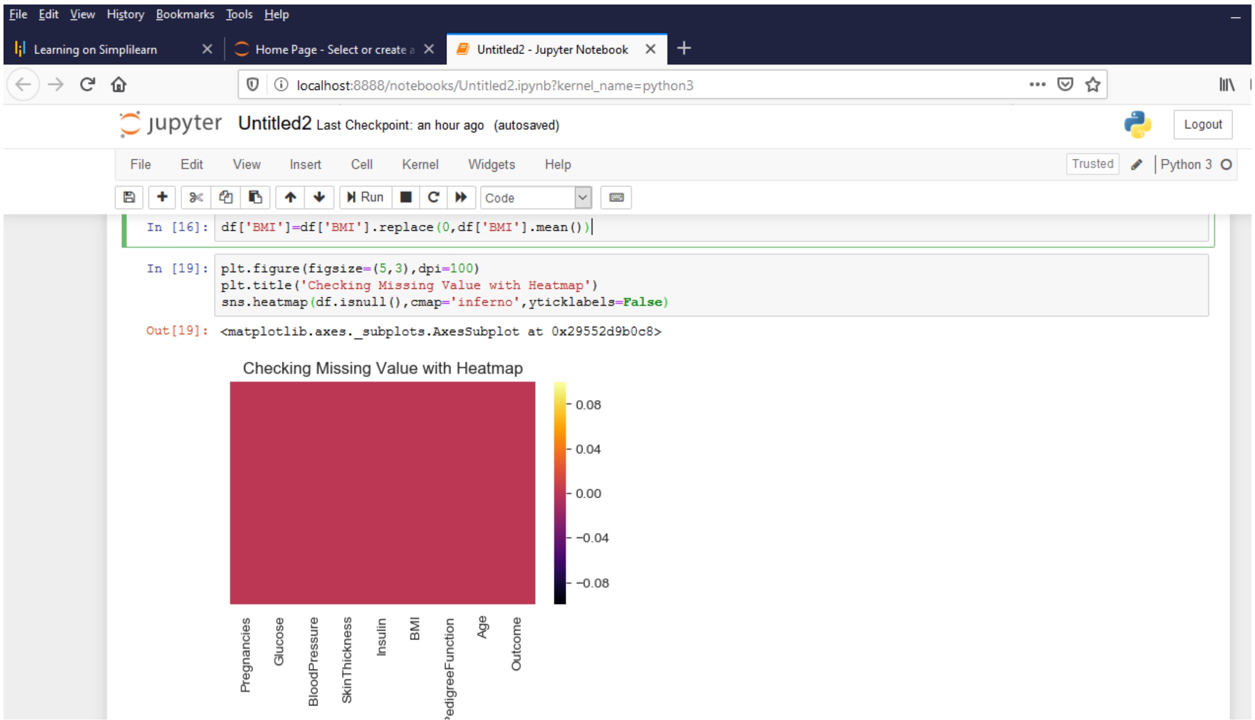
Code

df['BMI']=df['BMI'].replace(0,df['BMI'].mean())

plt.figure(figsize=(5,3),dpi=100)

plt.title('Checking Missing Value with Heatmap') sns.heatmap(df.isnull(),cmap='inferno',yticklabels=**False**)

Output

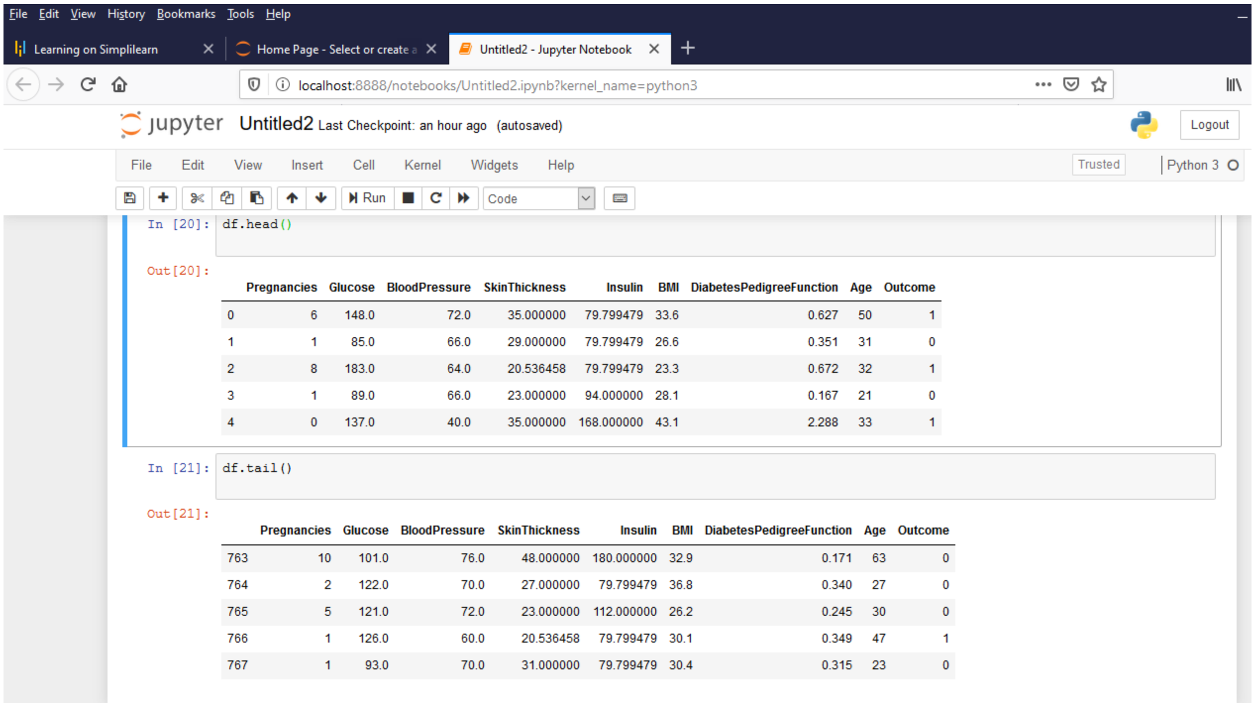


Code

df.head()

df.tail()

Output



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50	1
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	0
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32	1
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33	1

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101.0	76.0	48.000000	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.000000	79.799479	36.8	0.340	27	0
765	5	121.0	72.0	23.000000	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	20.536458	79.799479	30.1	0.349	47	1
767	1	93.0	70.0	31.000000	79.799479	30.4	0.315	23	0

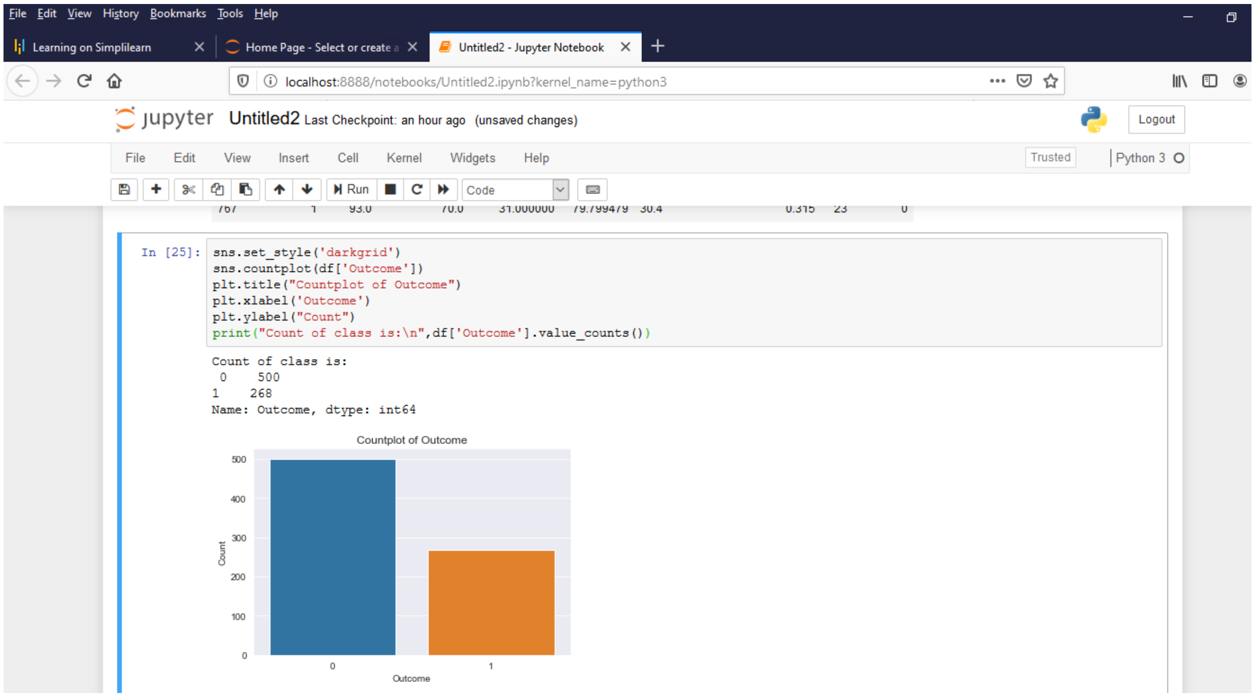
Project Task : Week 2

Countplot

Code

```
sns.set_style('darkgrid')

sns.countplot(df['Outcome'])

plt.title("Countplot of Outcome")

plt.xlabel('Outcome')

plt.ylabel("Count")

print("Count of class is:\n",df['Outcome'].value_counts())
```

Output



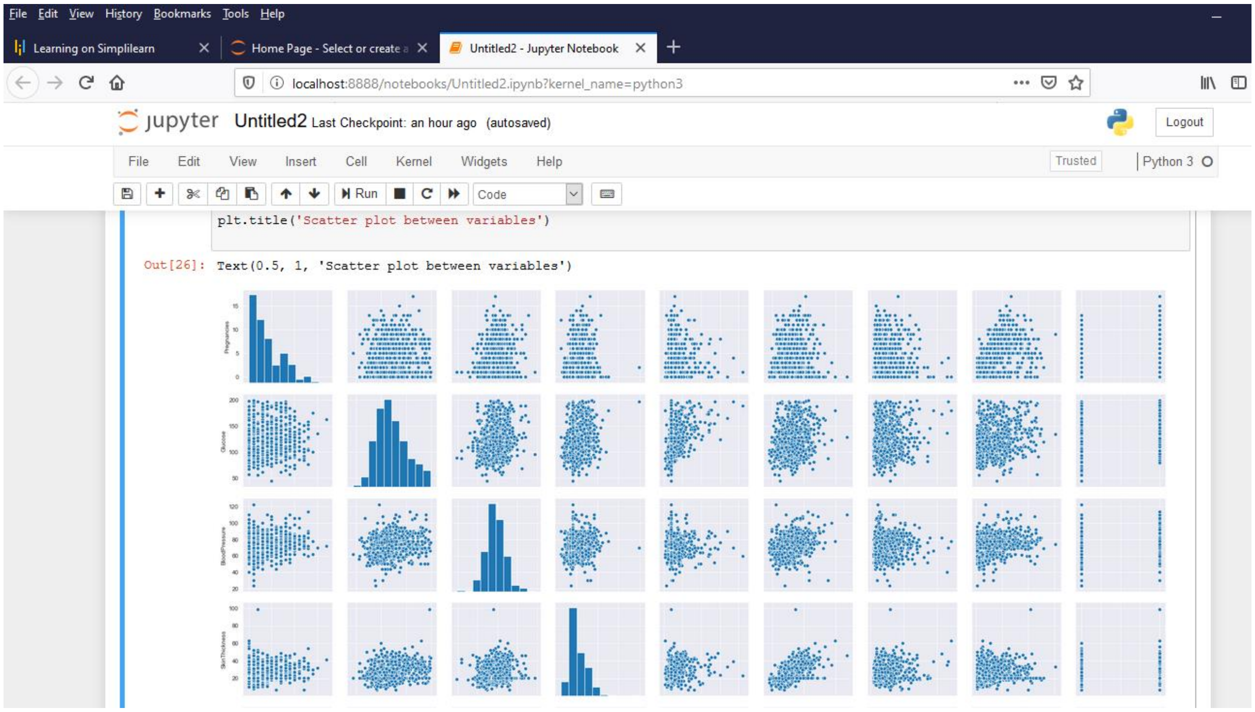
Creating Scatter Plot

Code

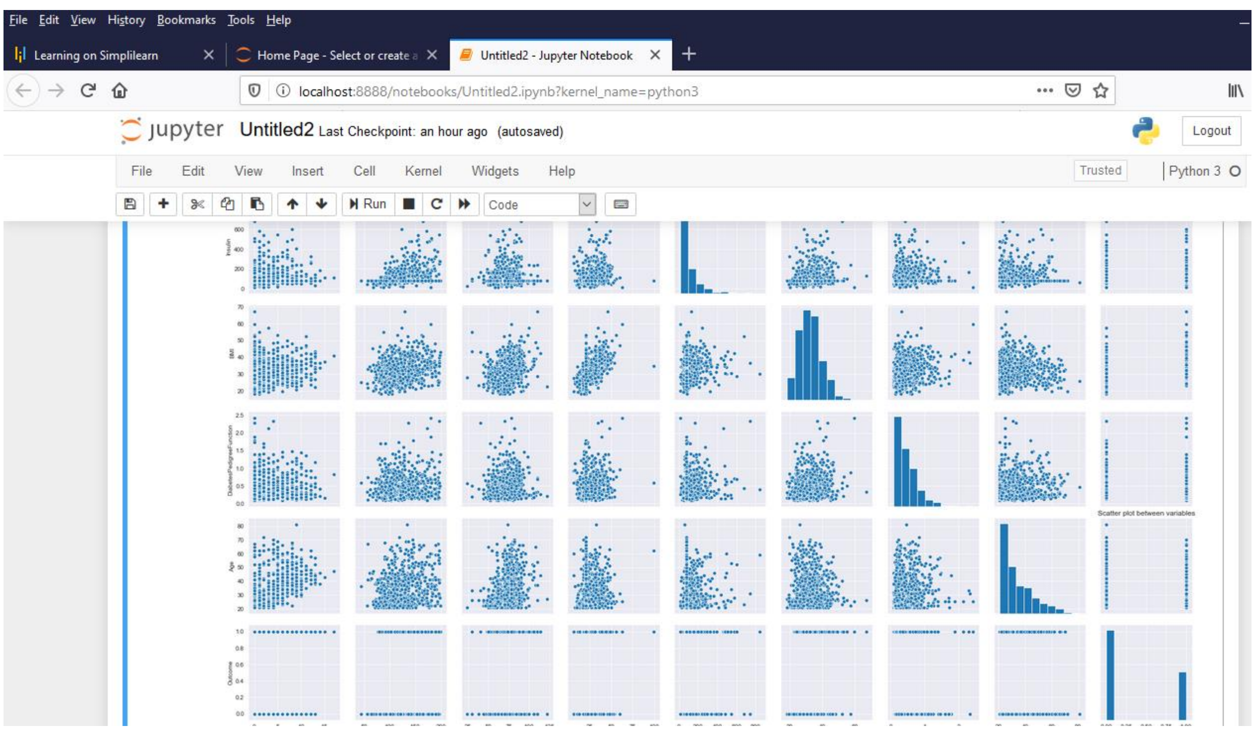
sns.pairplot(df)

plt.title('Scatter plot between variables')

Output



File Edit View History Bookmarks Tools Help
Learning on Simplilearn
Home Page - Select or create a
Untitled2 - Jupyter Notebook
localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3
jupyter Untitled2 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
plt.title('Scatter plot between variables')
Out[26]: Text(0.5, 1, 'Scatter plot between variables')



File Edit View History Bookmarks Tools Help
Learning on Simplilearn
Home Page - Select or create a
Untitled2 - Jupyter Notebook
localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3
jupyter Untitled2 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

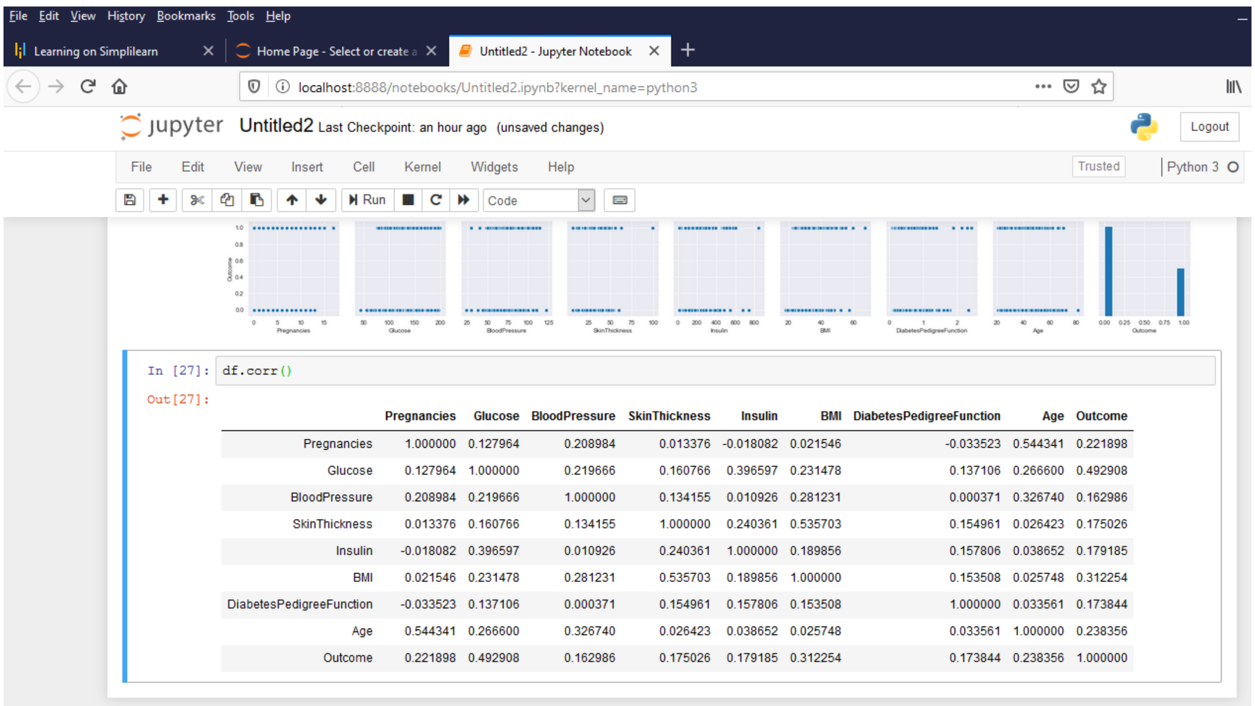
Skin Thickness and BMI , Pregnancies and Age has small positive Correlation.

Analyzing with Correlation

Code

df.corr()

Output



In [27]: df.corr()

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.018082	0.021546	-0.033523	0.544341	0.221898
Glucose	0.127964	1.000000	0.219666	0.160766	0.396597	0.231478	0.137106	0.266600	0.492908
BloodPressure	0.208984	0.219666	1.000000	0.134155	0.010926	0.281231	0.000371	0.326740	0.162986
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.240361	0.535703	0.154961	0.026423	0.175026
Insulin	-0.018082	0.396597	0.010926	0.240361	1.000000	0.189856	0.157806	0.038652	0.179185
BMI	0.021546	0.231478	0.281231	0.535703	0.189856	1.000000	0.153508	0.025748	0.312254
DiabetesPedigreeFunction	-0.033523	0.137106	0.000371	0.154961	0.157806	0.153508	1.000000	0.033561	0.173844
Age	0.544341	0.266600	0.326740	0.026423	0.038652	0.025748	0.033561	1.000000	0.238356
Outcome	0.221898	0.492908	0.162986	0.175026	0.179185	0.312254	0.173844	0.238356	1.000000

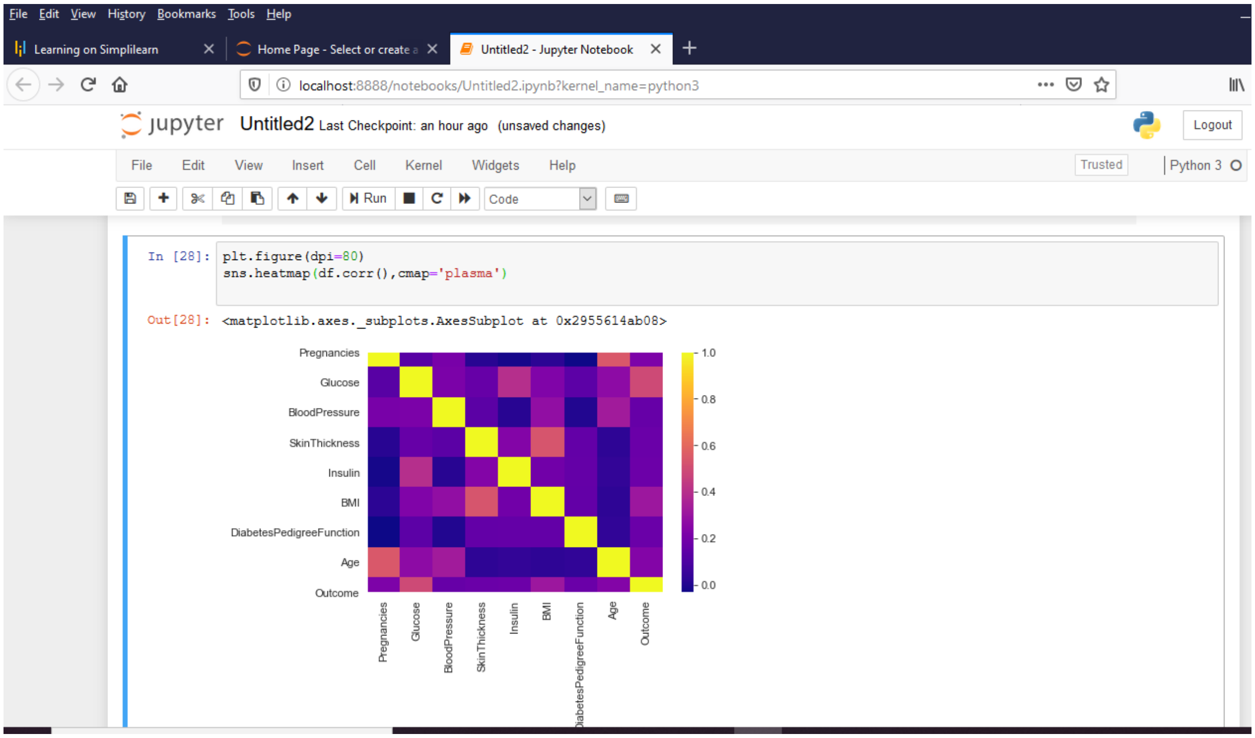
Skin Thickness and BMI , Pregnancies and Age has Strong positive correlation.

Code

plt.figure(dpi=80)

sns.heatmap(df.corr(),cmap='plasma')

Output



Project Task : Week 3

Preprocessing Data

Code

x=df.iloc[:,:-1].values

y=df.iloc[:,-1].values

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)

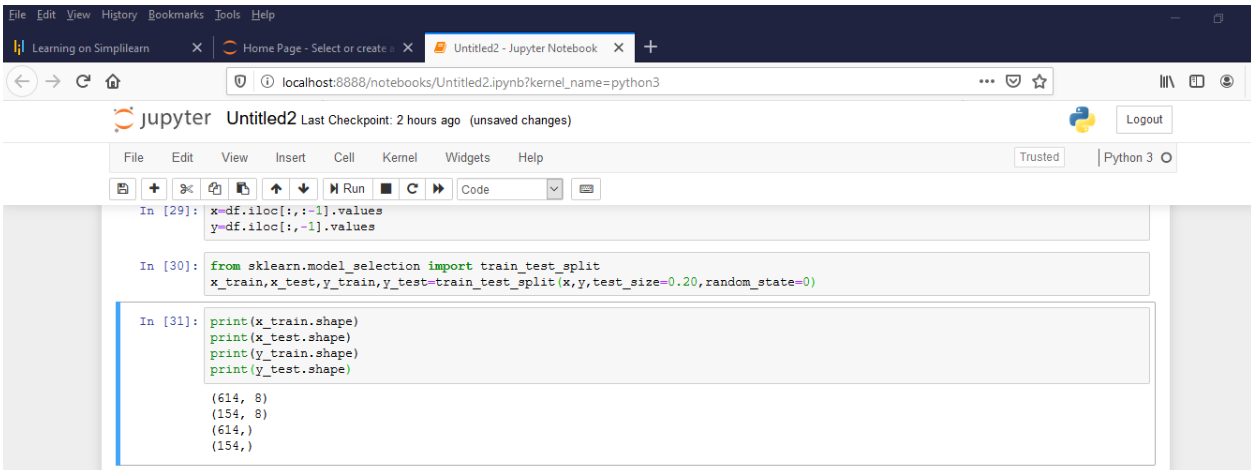
print(x_train.shape)

print(x_test.shape)

print(y_train.shape)

print(y_test.shape)

Output



Code

```
from sklearn.preprocessing import StandardScaler
Scale=StandardScaler()
x_train_std=Scale.fit_transform(x_train)
x_test_std=Scale.transform(x_test)
norm=lambda a:(a-min(a))/(max(a)-min(a))
df_norm=df.iloc[:,:-1]
df_normalized=df_norm.apply(norm)
```

x_train_norm,x_test_norm,y_train_norm,y_test_norm=train_test_split(df_normalized.values,y,test_size=0.20,random_state=0)

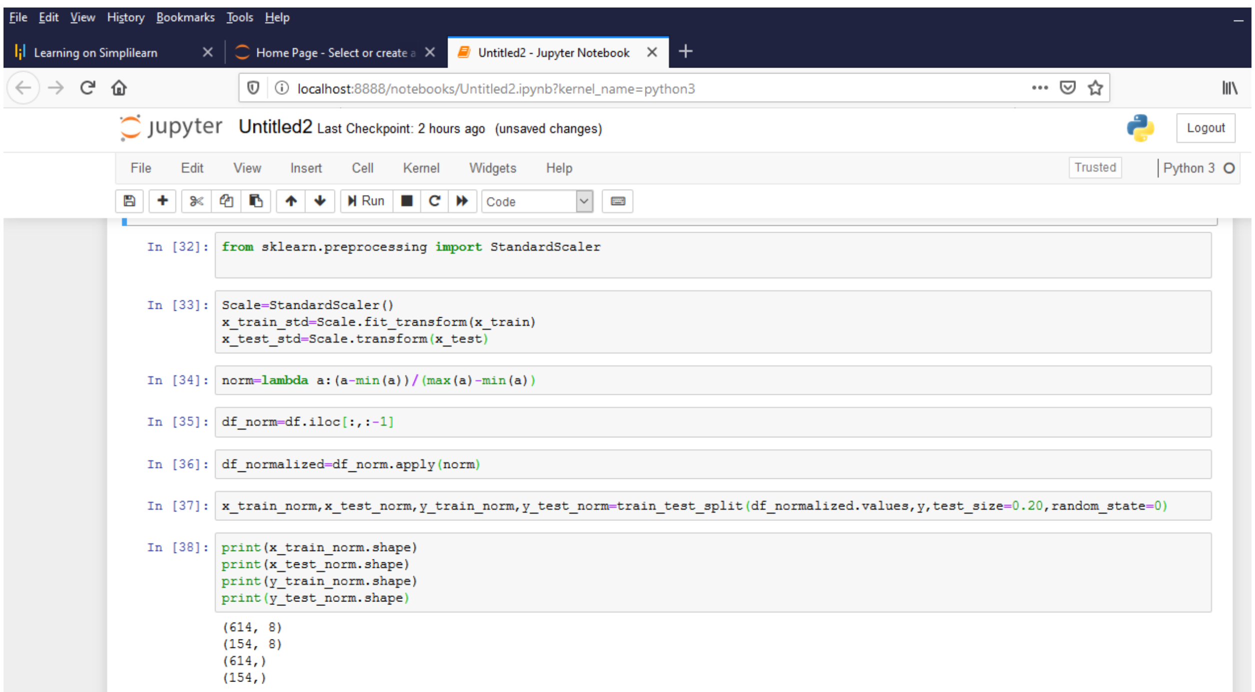
print(x_train_norm.shape)

print(x_test_norm.shape)

print(y_train_norm.shape)

print(y_test_norm.shape)

Output



Analyzing with Classification

KNN

Code

```
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=25)
knn_model.fit(x_train_std,y_train)
knn_pred=knn_model.predict(x_test_std)



print("Model Validation ==>\n")
print("Accuracy Score of KNN Model::")
print(metrics.accuracy_score(y_test,knn_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,knn_pred),'\n')
print("\n","ROC Curve")
knn_prob=knn_model.predict_proba(x_test_std)
knn_prob1=knn_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,knn_prob1)
roc_auc_knn=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
```

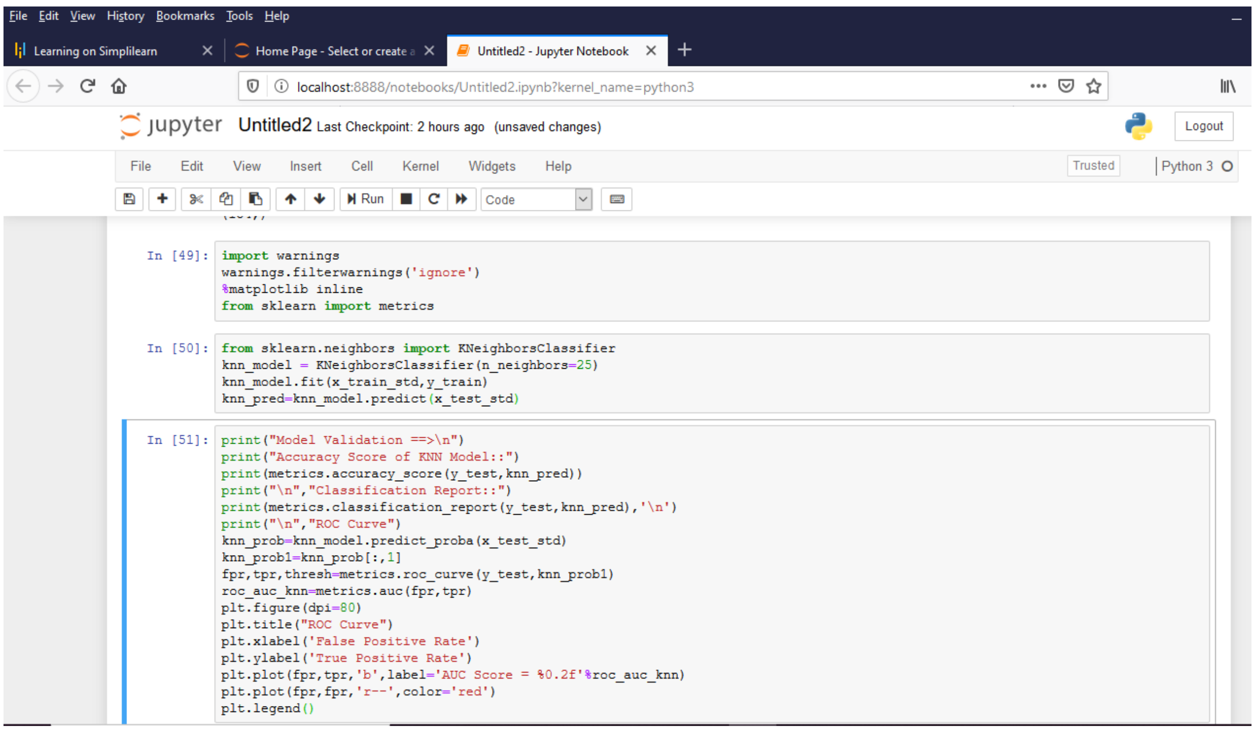
plt.ylabel('True Positive Rate')

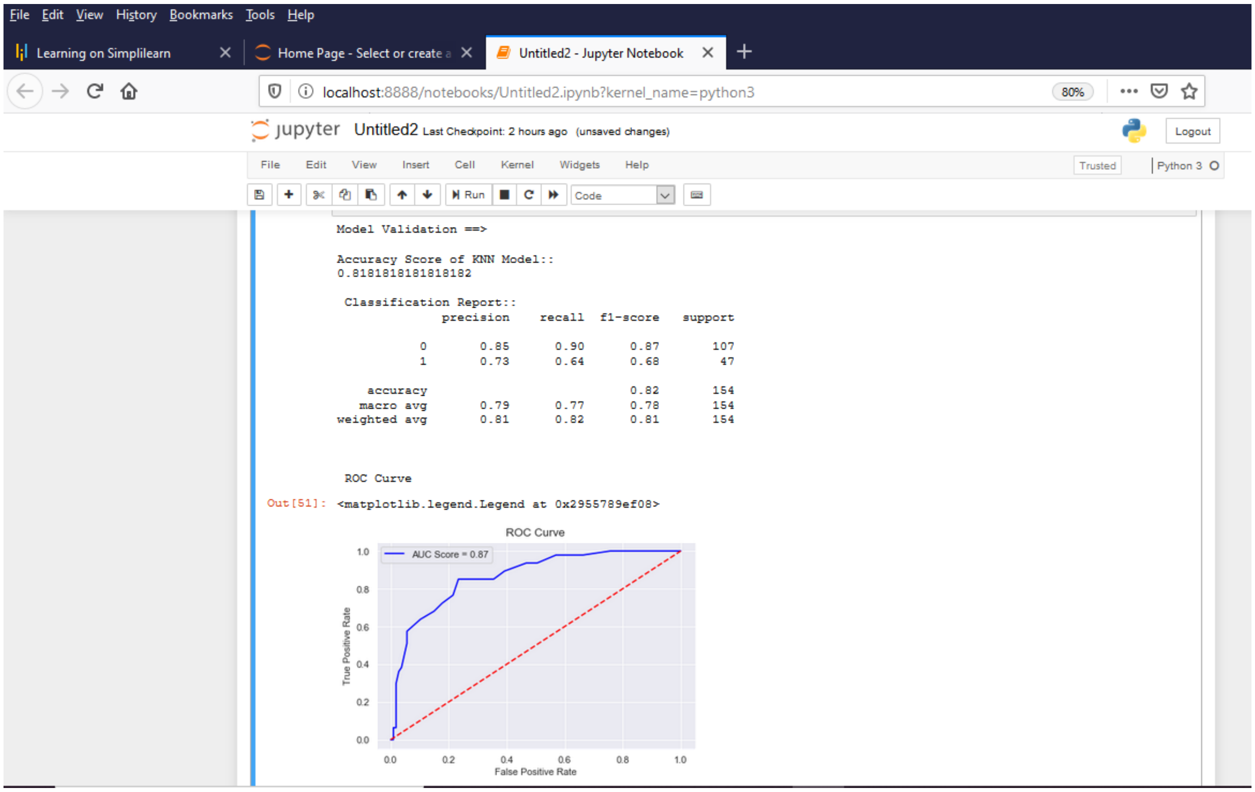
plt.plot(fpr,tpr,'b',label='AUC Score = **%0.2f'%roc_auc_knn**)

plt.plot(fpr,fpr,'r--',color='red')

plt.legend()

Output





KNN Normalization

Code

from sklearn.neighbors import KNeighborsClassifier

knn_model_norm = KNeighborsClassifier(n_neighbors=25)

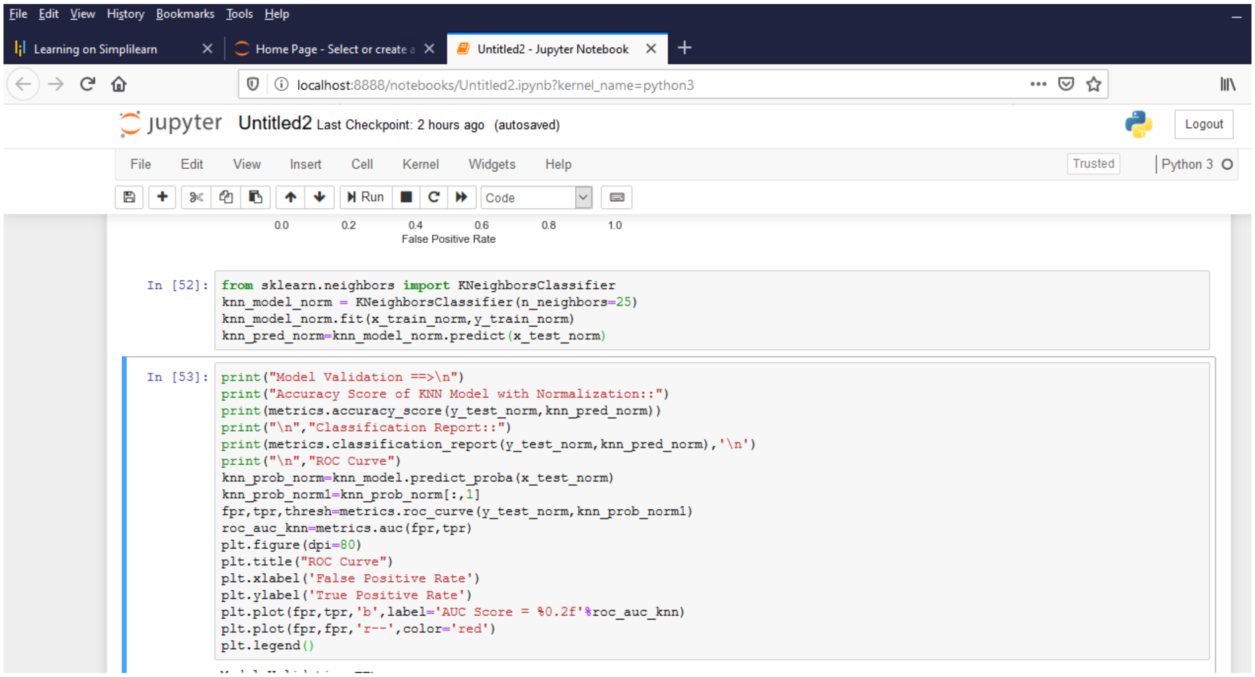
knn_model_norm.fit(x_train_norm,y_train_norm)

knn_pred_norm=knn_model_norm.predict(x_test_norm)

print("Model Validation ==>**\n**")

```
print("Accuracy Score of KNN Model with Normalization::")
print(metrics.accuracy_score(y_test_norm,knn_pred_norm))
print("\n","Classification Report::")
print(metrics.classification_report(y_test_norm,knn_pred_norm),'\n')
print("\n","ROC Curve")
knn_prob_norm=knn_model.predict_proba(x_test_norm)
knn_prob_norm1=knn_prob_norm[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test_norm,knn_prob_norm1)
roc_auc_knn=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_knn)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

Output




```
Model Validation ==>

Accuracy Score of KNN Model with Normalization::
0.8311688311688312

 Classification Report::
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       107
           1       0.74      0.68      0.71        47

    accuracy                           0.83       154
   macro avg       0.80      0.79      0.80       154
weighted avg       0.83      0.83      0.83       154
```

File Edit View History Bookmarks Tools Help

Learning on Simplilearn | Home Page - Select or create a | Untitled2 - Jupyter Notebook

localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3

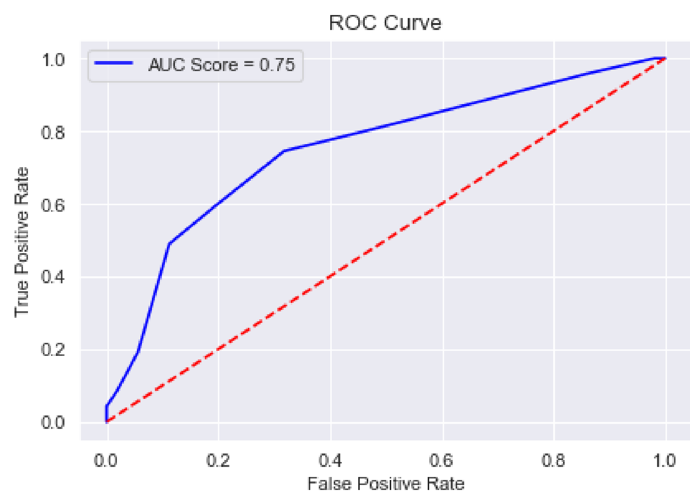
jupyter Untitled2 Last Checkpoint: 2 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

```
    accuracy                           0.83       154
   macro avg       0.80      0.79      0.80       154
weighted avg       0.83      0.83      0.83       154


 ROC Curve
```

Out[53]: <matplotlib.legend.Legend at 0x295578cdf88>



Support Vector

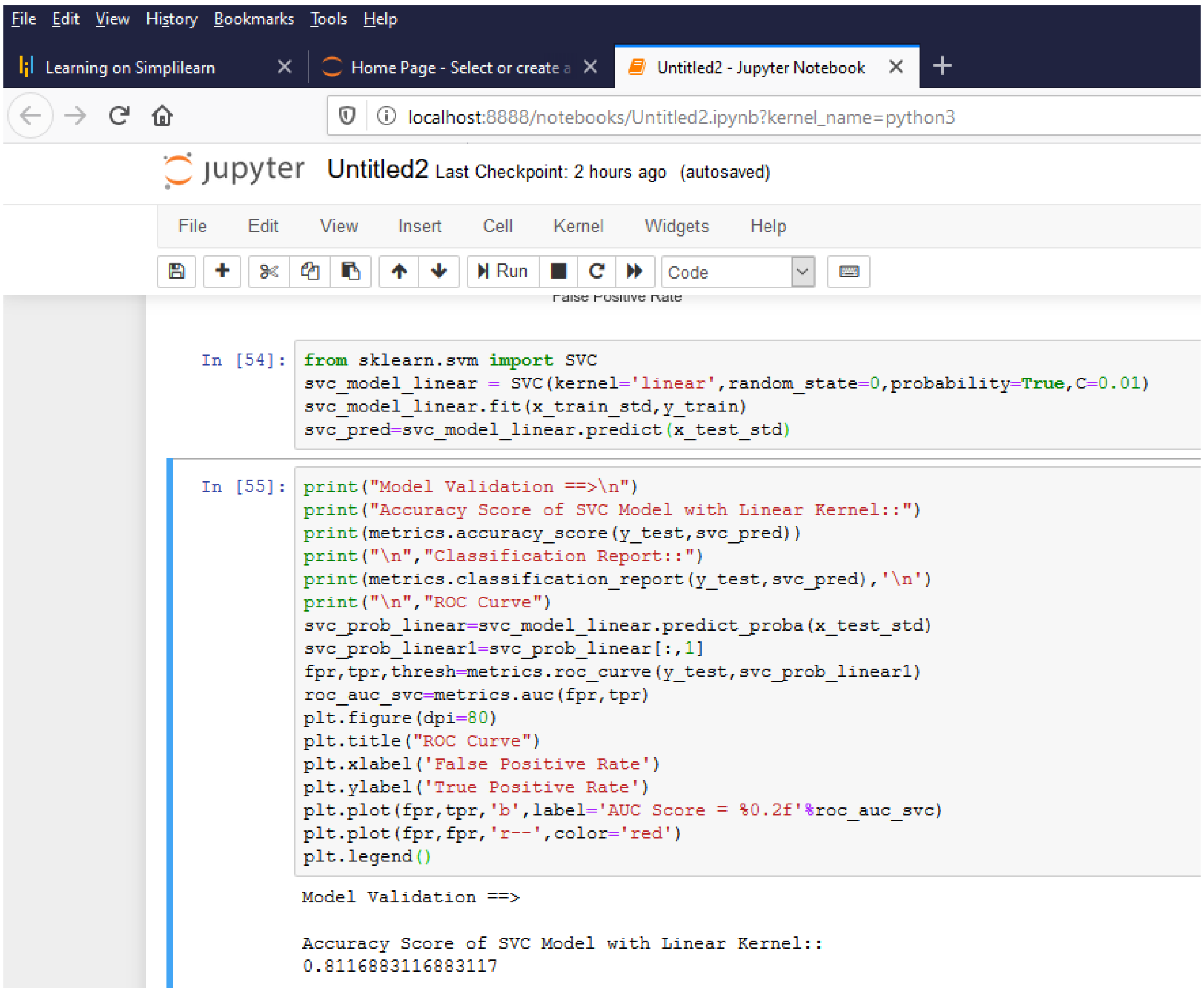
Code

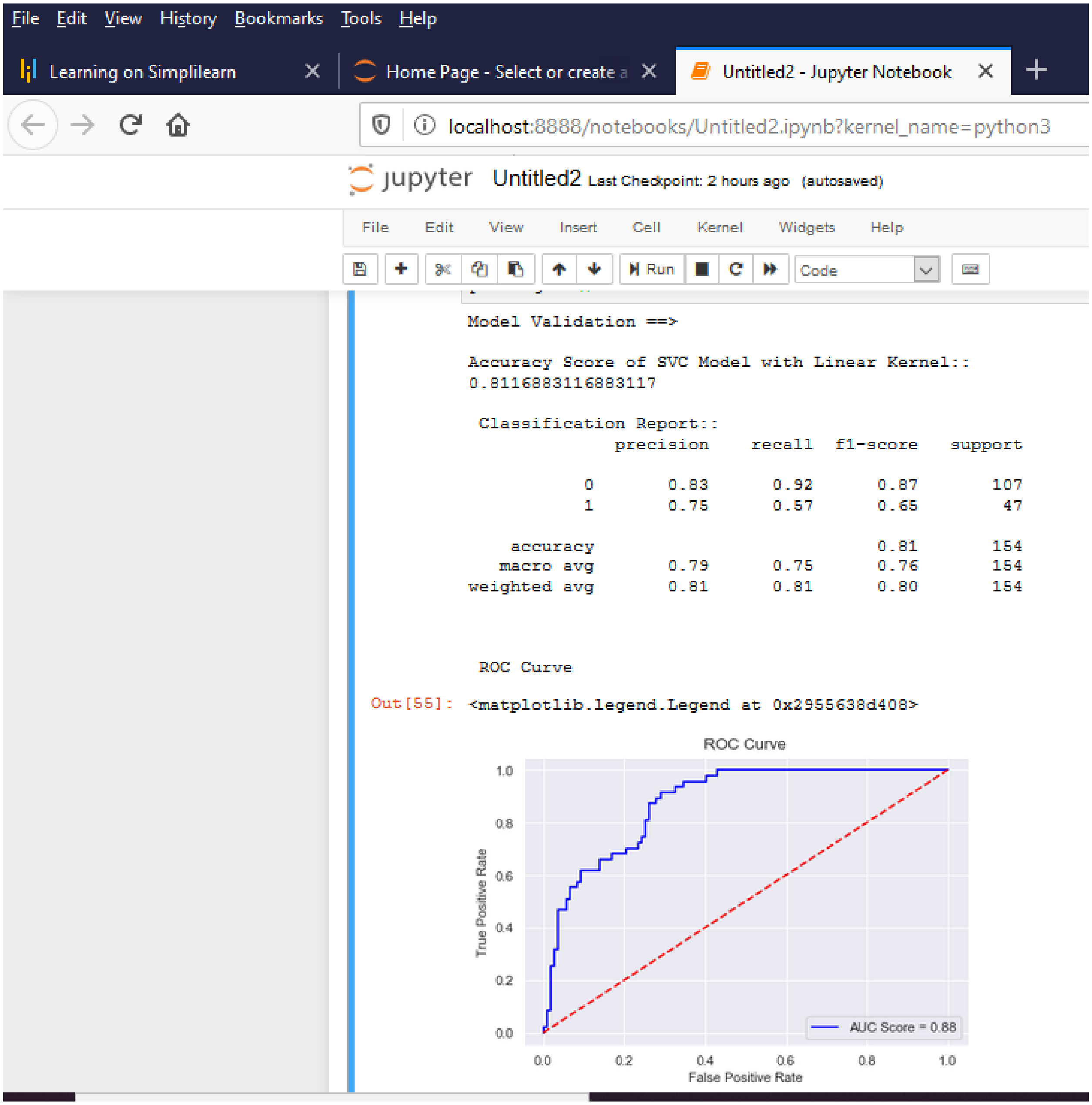
```
from sklearn.svm import SVC
svc_model_linear = SVC(kernel='linear',random_state=0,probability=True,C=0.01)
svc_model_linear.fit(x_train_std,y_train)
svc_pred=svc_model_linear.predict(x_test_std)



print("Model Validation ==>\n")
print("Accuracy Score of SVC Model with Linear Kernel::")
print(metrics.accuracy_score(y_test,svc_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,svc_pred),'\n')
print("\n","ROC Curve")
svc_prob_linear=svc_model_linear.predict_proba(x_test_std)
svc_prob_linear1=svc_prob_linear[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,svc_prob_linear1)
roc_auc_svc=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
```

plt.plot(fpr,fpr,'r--',color='red')

plt.legend()

Output






```
Model Validation ==>

Accuracy Score of SVC Model with Linear Kernel::
0.8116883116883117

 Classification Report::
              precision    recall  f1-score   support

           0       0.83      0.92      0.87       107
           1       0.75      0.57      0.65        47

    accuracy                           0.81       154
   macro avg       0.79      0.75      0.76       154
weighted avg       0.81      0.81      0.80       154


 ROC Curve
Out[55]: <matplotlib.legend.Legend at 0x2955638d408>
```

Code

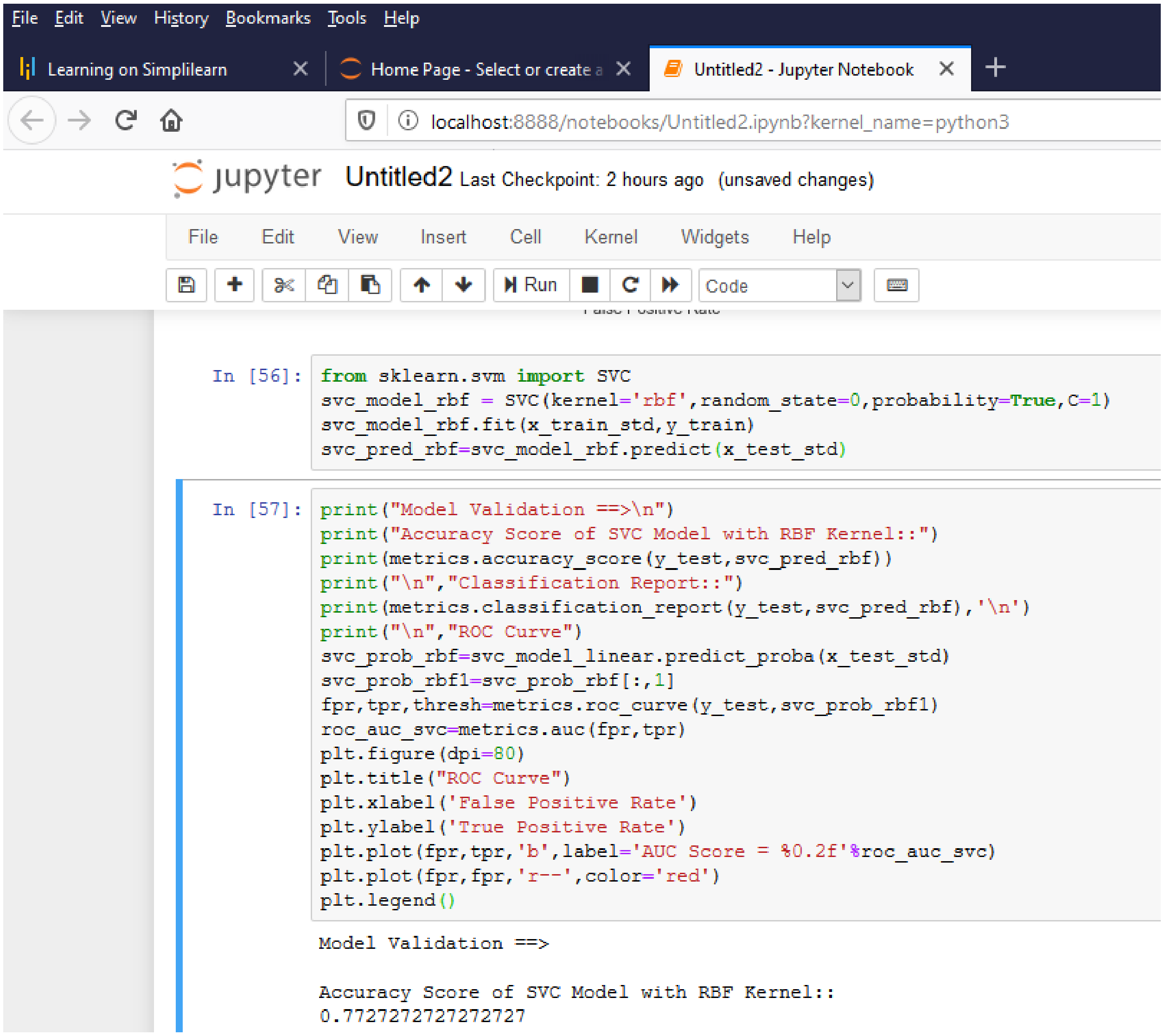
from sklearn.svm import SVC

svc_model_rbf = SVC(kernel='rbf',random_state=0,probability=**True**,C=1)

svc_model_rbf.fit(x_train_std,y_train)

```
svc_pred_rbf=svc_model_rbf.predict(x_test_std)


print("Model Validation ==>\n")
print("Accuracy Score of SVC Model with RBF Kernel::")
print(metrics.accuracy_score(y_test,svc_pred_rbf))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,svc_pred_rbf),'\n')
print("\n","ROC Curve")
svc_prob_rbf=svc_model_linear.predict_proba(x_test_std)
svc_prob_rbf1=svc_prob_rbf[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,svc_prob_rbf1)
roc_auc_svc=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

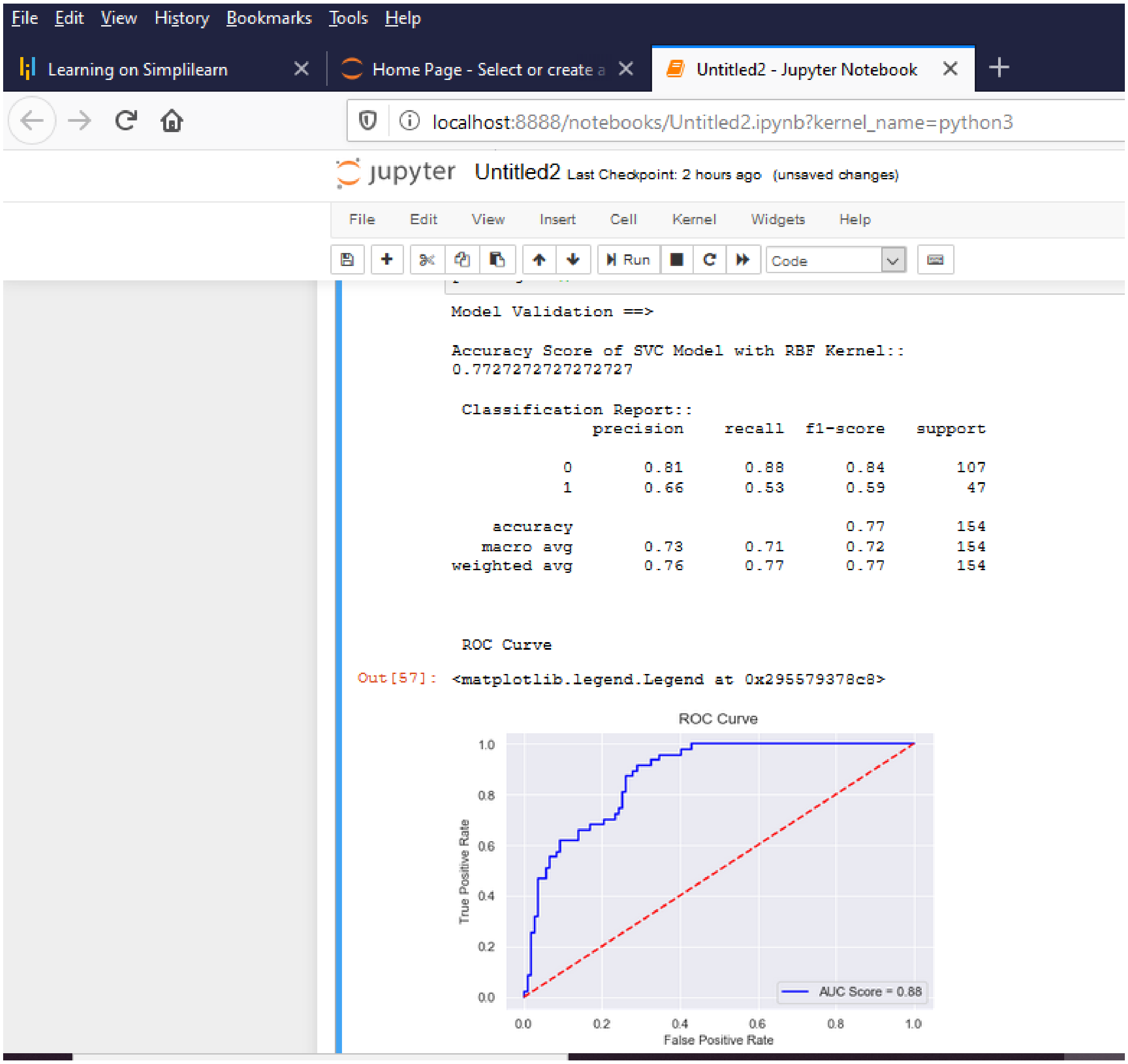
Output



```
In [56]: from sklearn.svm import SVC
         svc_model_rbf = SVC(kernel='rbf',random_state=0,probability=True,C=1)
         svc_model_rbf.fit(x_train_std,y_train)
         svc_pred_rbf=svc_model_rbf.predict(x_test_std)
```

```
In [57]: print("Model Validation ==>\n")
         print("Accuracy Score of SVC Model with RBF Kernel::")
         print(metrics.accuracy_score(y_test,svc_pred_rbf))
         print("\n","Classification Report::")
         print(metrics.classification_report(y_test,svc_pred_rbf),'\n')
         print("\n","ROC Curve")
         svc_prob_rbf=svc_model_linear.predict_proba(x_test_std)
         svc_prob_rbf1=svc_prob_rbf[:,1]
         fpr,tpr,thresh=metrics.roc_curve(y_test,svc_prob_rbf1)
         roc_auc_svc=metrics.auc(fpr,tpr)
         plt.figure(dpi=80)
         plt.title("ROC Curve")
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_svc)
         plt.plot(fpr,fpr,'r--',color='red')
         plt.legend()
```

```
Model Validation ==>

Accuracy Score of SVC Model with RBF Kernel::
0.7727272727272727
```



Comparing SVC with KNN , SVC with c=0.01 is good in terms of AUC score

Logistic Regression

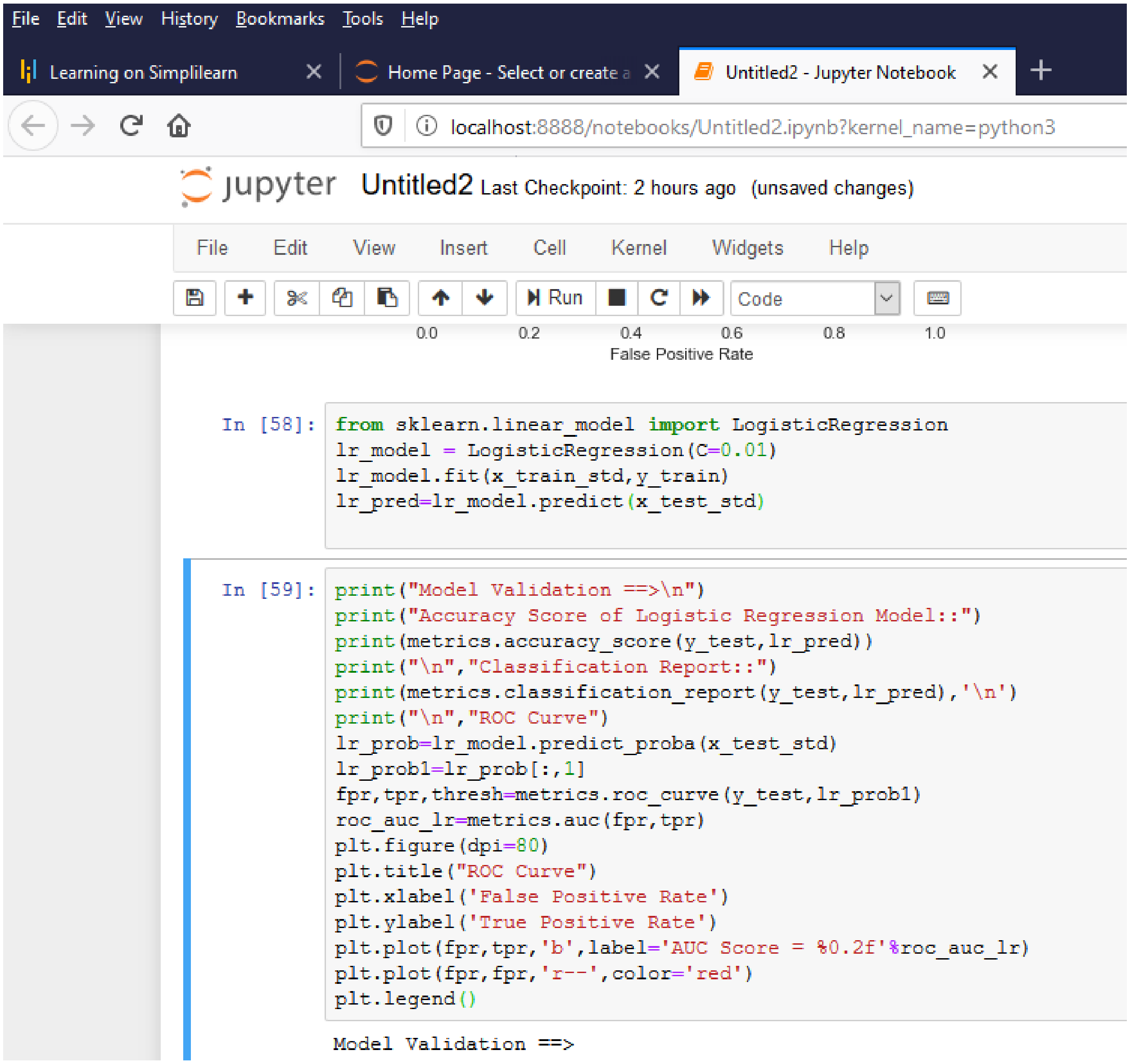
Code

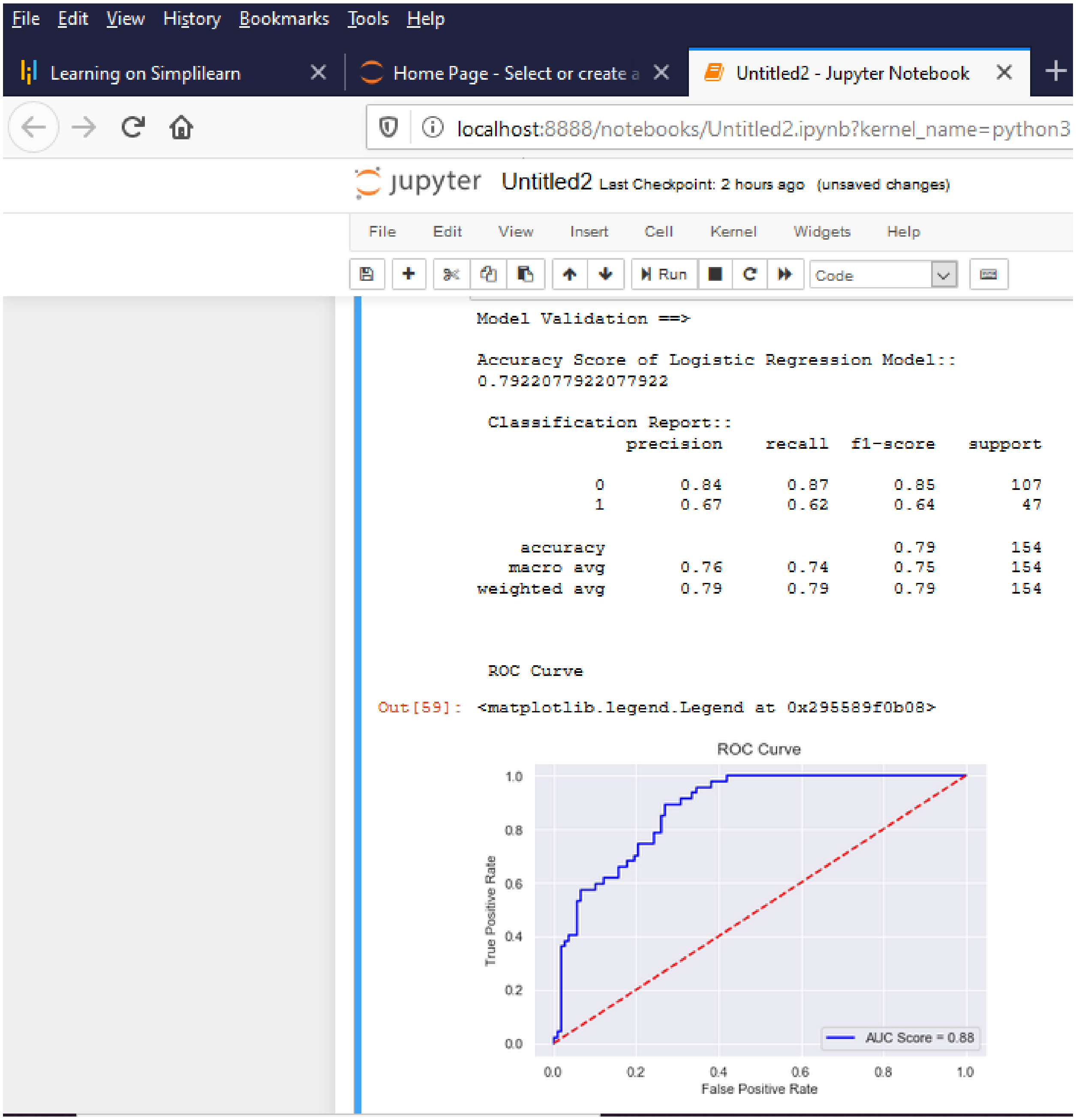
from sklearn.linear_model import LogisticRegression

lr_model = LogisticRegression(C=0.01)

```
lr_model.fit(x_train_std,y_train)
lr_pred=lr_model.predict(x_test_std)



print("Model Validation ==>\n")
print("Accuracy Score of Logistic Regression Model::")
print(metrics.accuracy_score(y_test,lr_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,lr_pred),'\n')
print("\n","ROC Curve")
lr_prob=lr_model.predict_proba(x_test_std)
lr_prob1=lr_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,lr_prob1)
roc_auc_lr=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_lr)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
```

Output


File Edit View History Bookmarks Tools Help
Learning on Simplilearn
Home Page - Select or create a
Untitled2 - Jupyter Notebook
localhost:8888/notebooks/Untitled2.ipynb?kernel_name=python3
jupyter Untitled2 Last Checkpoint: 2 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
0.0 0.2 0.4 0.6 0.8 1.0
False Positive Rate
In [58]:
from sklearn.linear_model import LogisticRegression
lr_model = LogisticRegression(C=0.01)
lr_model.fit(x_train_std,y_train)
lr_pred=lr_model.predict(x_test_std)
In [59]:
print("Model Validation ==>\n")
print("Accuracy Score of Logistic Regression Model::")
print(metrics.accuracy_score(y_test,lr_pred))
print("\n","Classification Report::")
print(metrics.classification_report(y_test,lr_pred),'\n')
print("\n","ROC Curve")
lr_prob=lr_model.predict_proba(x_test_std)
lr_prob1=lr_prob[:,1]
fpr,tpr,thresh=metrics.roc_curve(y_test,lr_prob1)
roc_auc_lr=metrics.auc(fpr,tpr)
plt.figure(dpi=80)
plt.title("ROC Curve")
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_lr)
plt.plot(fpr,fpr,'r--',color='red')
plt.legend()
Model Validation ==>



AUC score of Logistic Regression is better and Accuracy of KNN is better

Random Forest Classifier

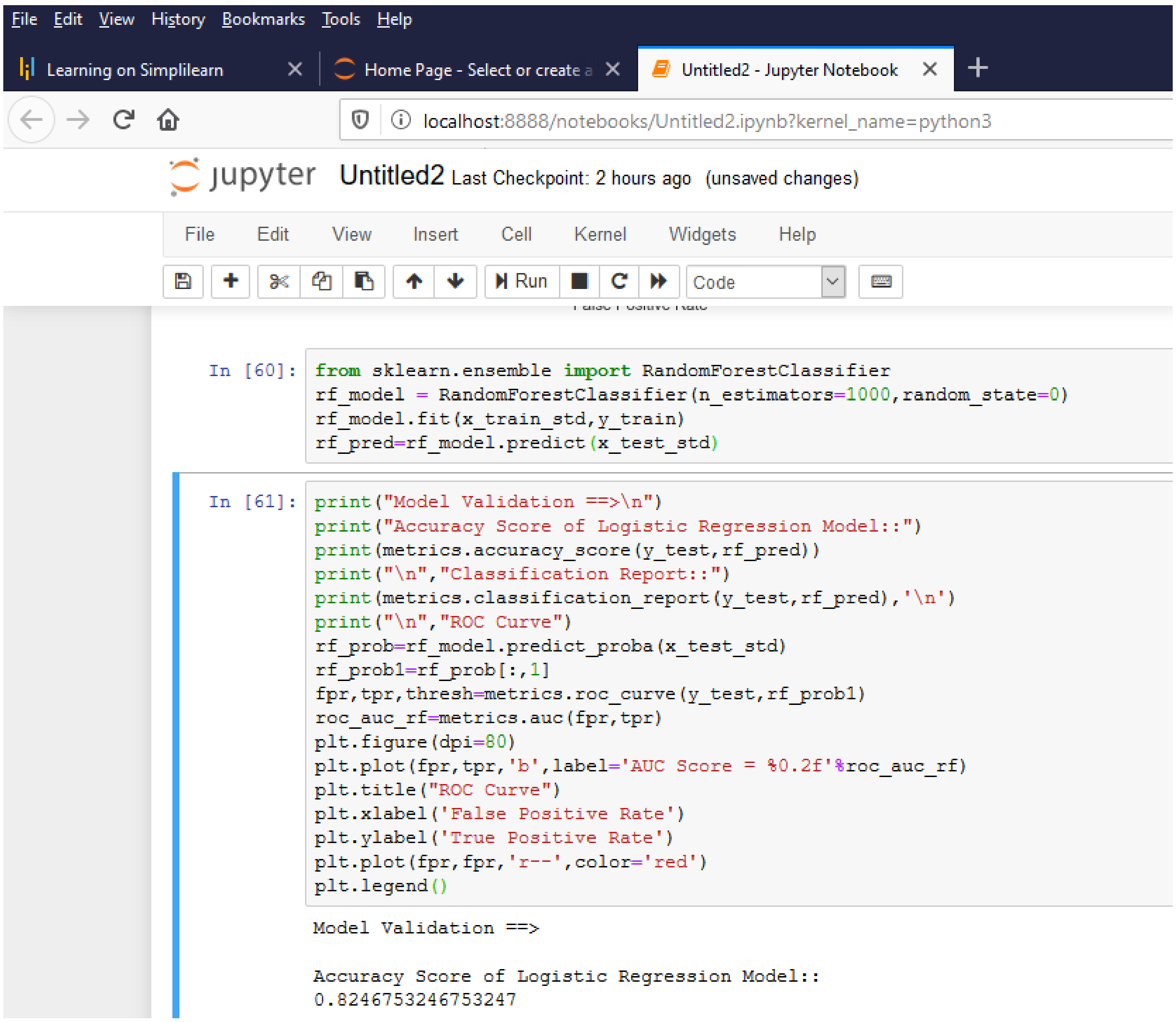
Code

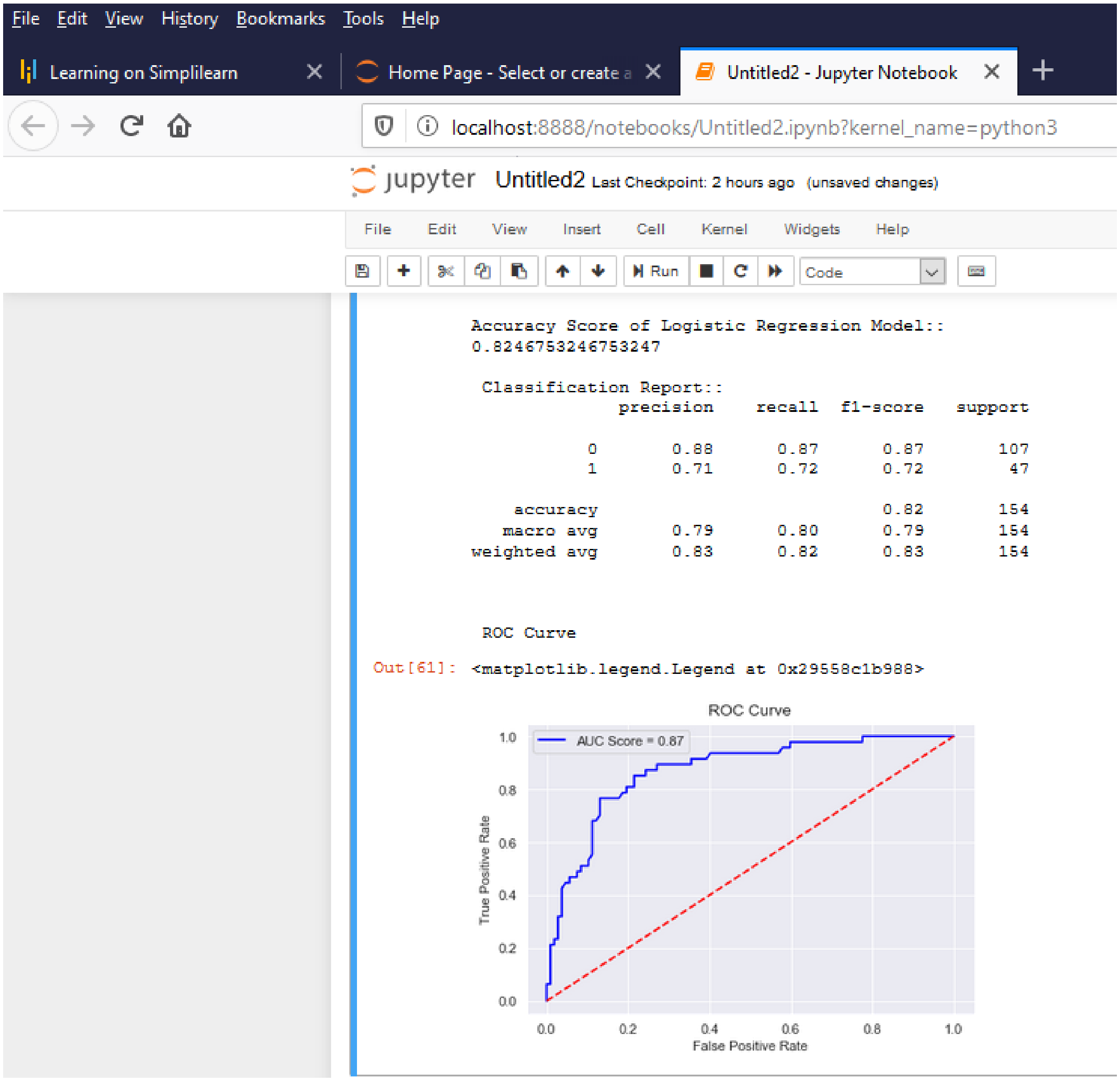
```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=1000,random_state=0)

rf_model.fit(x_train_std,y_train)

rf_pred=rf_model.predict(x_test_std)



print("Model Validation ==>\n")

print("Accuracy Score of Logistic Regression Model::")

print(metrics.accuracy_score(y_test,rf_pred))

print("\n","Classification Report::")

print(metrics.classification_report(y_test,rf_pred),'\n')

print("\n","ROC Curve")

rf_prob=rf_model.predict_proba(x_test_std)

rf_prob1=rf_prob[:,1]

fpr,tpr,thresh=metrics.roc_curve(y_test,rf_prob1)

roc_auc_rf=metrics.auc(fpr,tpr)

plt.figure(dpi=80)

plt.plot(fpr,tpr,'b',label='AUC Score = %0.2f'%roc_auc_rf)

plt.title("ROC Curve")

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')
```

plt.plot(fpr,fpr,'r--',color='red')

plt.legend()

Output





Random Forest Classification is better , we can consider a loss in AUC by 1

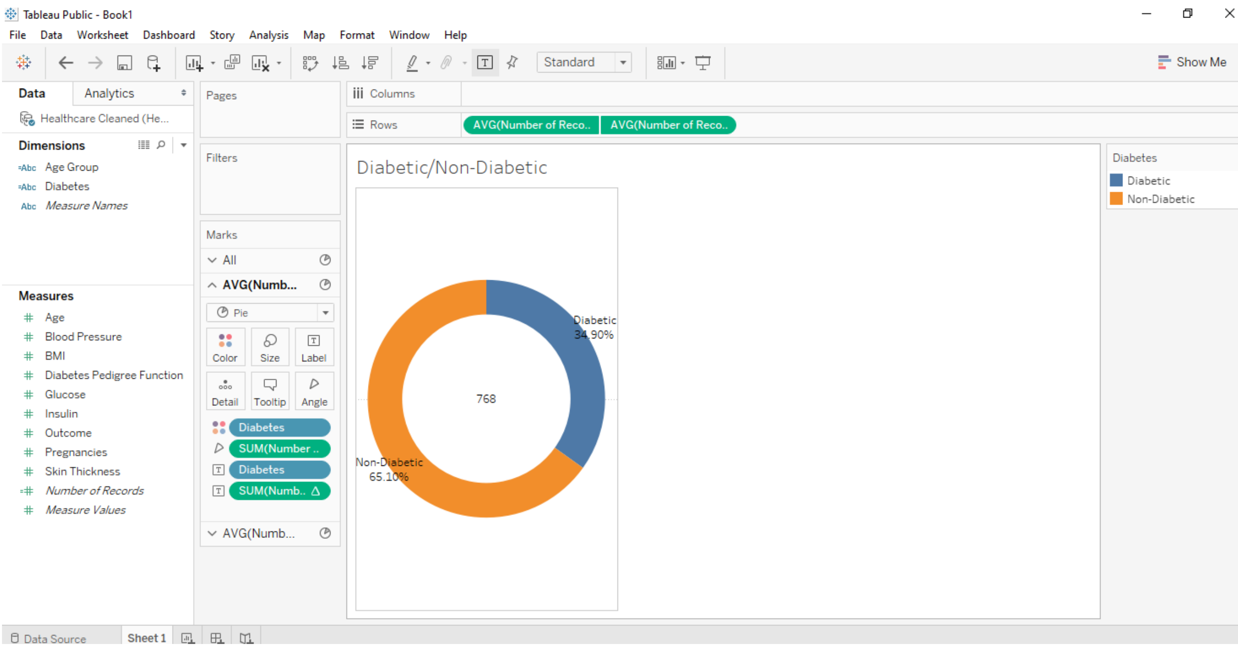
As html till week 3 :

file:///C:/Users/HOME/Downloads/DS%20Capstone%20-%20Health%20Care%20-%20KeerthiRajan.html

Project Task : Week 4

Visualization

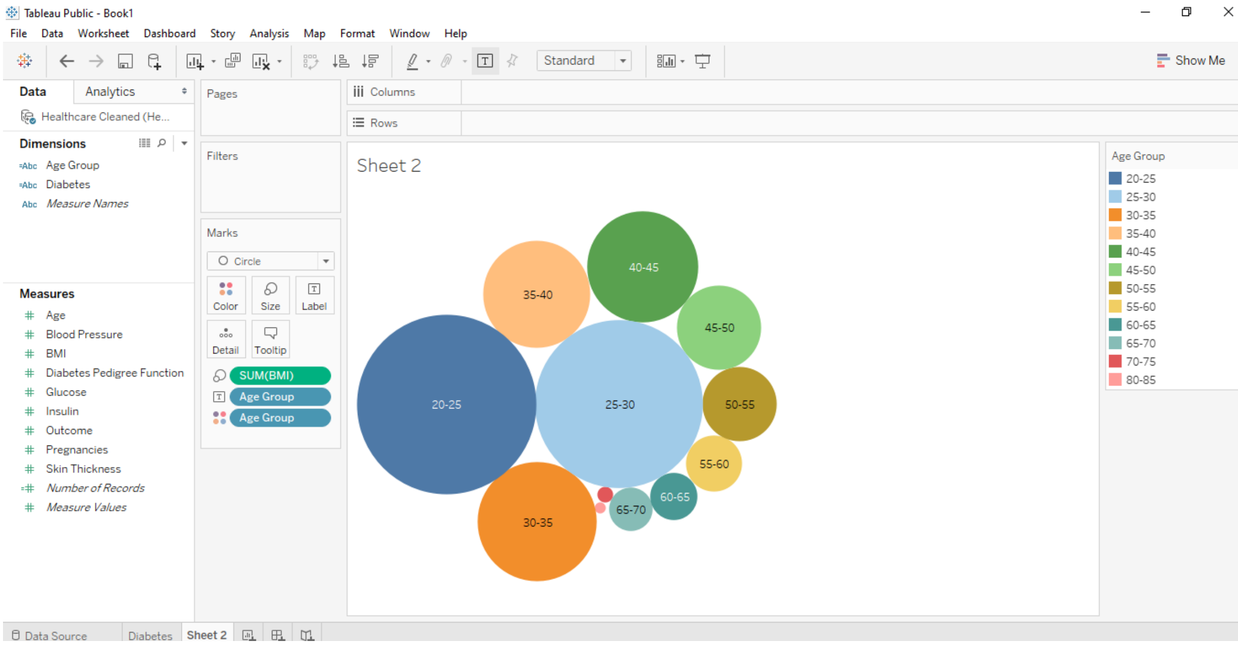
Diabetic and Non-Diabetic



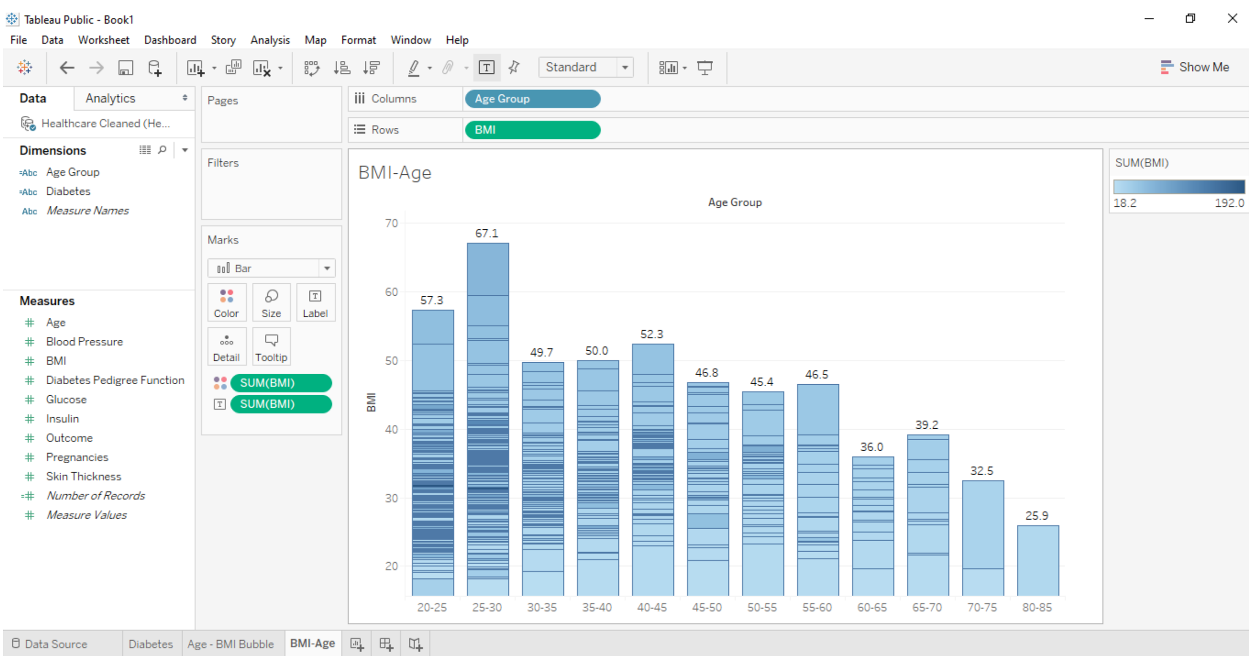
Diabetic – 65.10 %

Non-Diabetic – 34.90 %

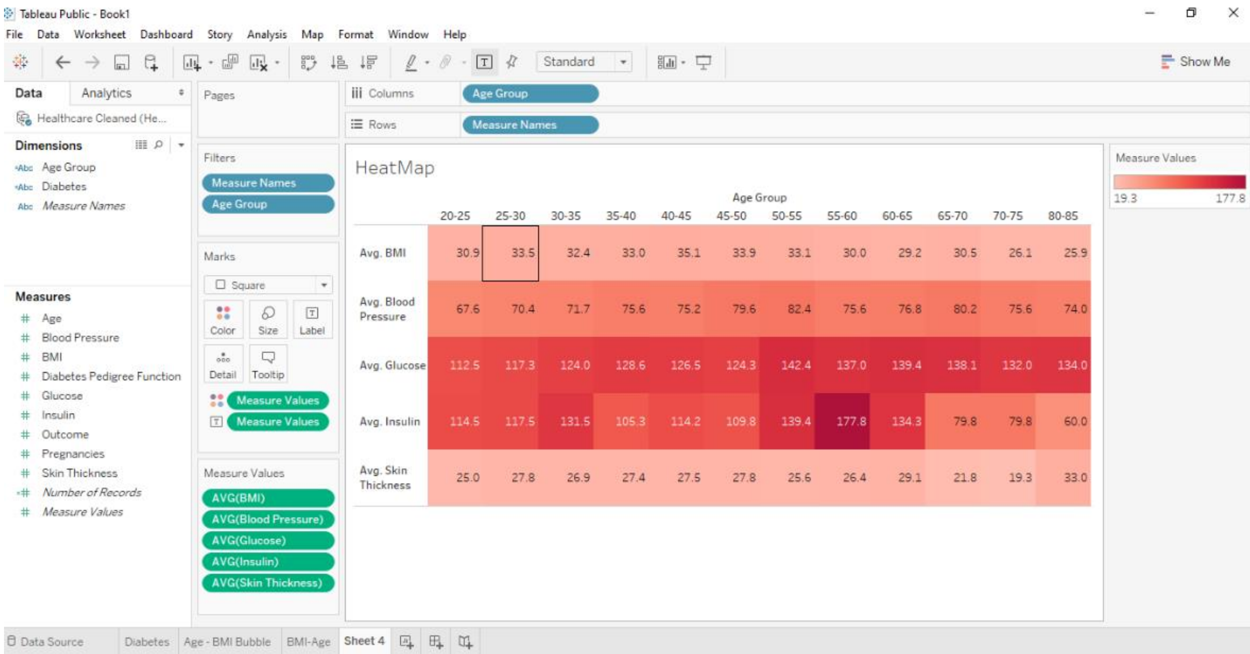
Blood Pressure Report



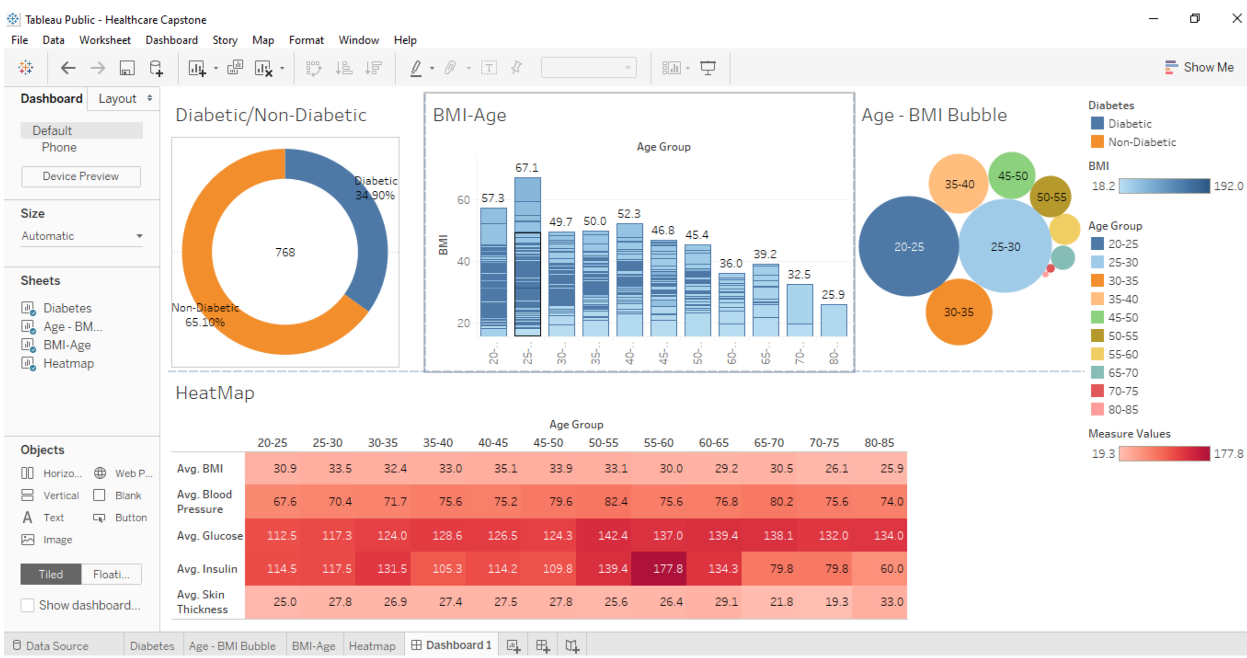
Body Mass Index to Age



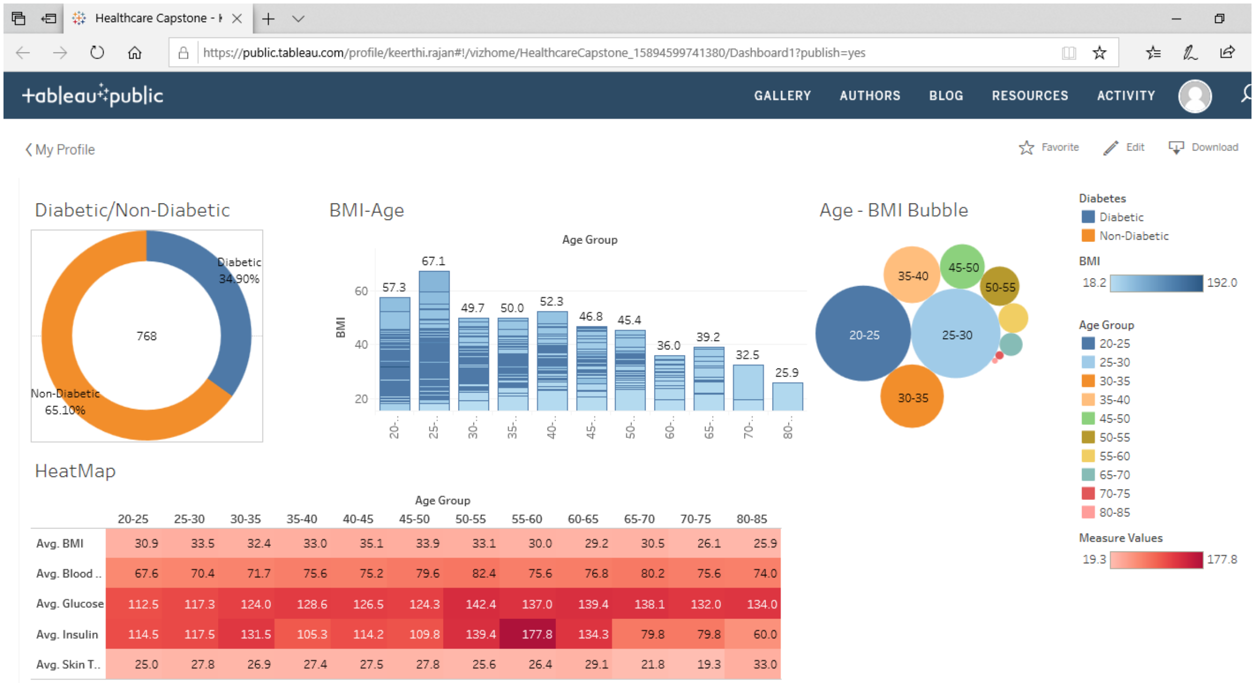
Heatmap



Dashboard



Final Dashboard from Tableau Public



URL :
https://public.tableau.com/profile/keerthi.rajan#!/vizhome/HealthcareCapstone_15894599741380/Dashboard1?publish=yes