SURUTHI S

225229141

PML LAB 6

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
df = pd.read_csv("diabetes.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
df.sample(6)
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
50	1	103	80	11	82	19.4	0.491	22	0
469	6	154	78	41	140	46.1	0.571	27	0
260	3	191	68	15	130	30.9	0.299	34	0
427	1	181	64	30	180	34.1	0.328	38	1
742	1	109	58	18	116	28.5	0.219	22	0
754	8	154	78	32	0	32.4	0.443	45	1

In [6]:
```
df.shape
```

Out[6]:
```
(768, 9)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [8]:
```
df.dtypes
```

Out[8]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

In [9]:
```
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 [10]:
```
df.value_counts()
```

Out[10]:
```
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI   DiabetesPedigreeFunction  Age  Outcome
0            57       60             0              0        21.7  0.735                     67   0          1
             67       76             0              0        45.3  0.194                     46   0          1
5            103      108            37             0        39.2  0.305                     65   0          1
             104      74             0              0        28.8  0.153                     48   0          1
             105      72             29             325      36.9  0.159                     28   0          1
                                                                                                        ..
2            84       50             23             76       30.4  0.968                     21   0          1
             85       65             0              0        39.6  0.930                     27   0          1
             87       0              23             0        28.9  0.773                     25   0          1
                      58             16             52       32.7  0.166                     25   0          1
17           163      72             41             114      40.9  0.817                     47   1          1
Length: 768, dtype: int64
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

STEP 2

IDENTIFYING RELATIONSHIPS

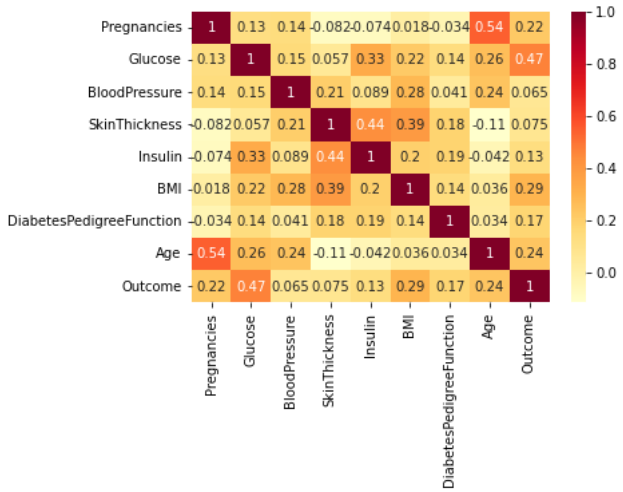
In [12]:
```
import seaborn as sns
df.corr()
```

Out[12]:

	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 [13]:
```
sns.heatmap(df.corr(),annot=True,cmap='YlOrRd')
```

Out[13]:
```
<AxesSubplot:>
```



STEP 3 -PREDICTION USING ONE FEATURE

In [14]:
```
from sklearn.linear_model import LogisticRegression
from  sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

In [15]:
```
df.Outcome.value_counts()
```

Out[15]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [16]:
```
# stratified shuffle split

from sklearn.model_selection import StratifiedShuffleSplit

sss = StratifiedShuffleSplit(n_splits=4,test_size = 0.30,random_state = 42)


```

In [17]:
```
features = df[['Age']]
labels = df[['Outcome']]

for train_index,test_index in sss.split(features,labels):
    x_train,x_test = features.iloc[train_index],features.iloc[test_index]
    y_train,y_test = labels.iloc[train_index],labels.iloc[test_index]
```

In [18]:
```
x_train.shape
```

Out[18]: (537, 1)

In [19]:
```
x_test.shape
```

Out[19]: (231, 1)

In [20]:
```
logreg = LogisticRegression()
logreg.fit(x_train,y_train)
y_pred = logreg.predict(x_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.93      0.78       150
           1       0.52      0.15      0.23        81

    accuracy                           0.65       231
   macro avg       0.60      0.54      0.50       231
weighted avg       0.62      0.65      0.59       231


C:\Users\SURUTHI S\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

In [21]:
```
from sklearn.metrics import accuracy_score

accuracy_score(y_test,y_pred)
```

Out[21]: 0.6536796536796536

In [22]:
```
logreg.coef_
```

Out[22]: array([[0.03700533]])

In [23]:
```
logreg.intercept_
```

Out[23]: array([-1.87818016])

In [24]:
```
# age = 60 , outcome = ?
age = 60
lrf = logreg.coef_*age + logreg.intercept_
```

In [25]:
```
lrf
```

Out[25]: array([[0.34213957]])

In [26]:
```
from scipy.special import expit

expit(lrf)

#Expit (a.k.a. logistic sigmoid) ufunc for ndarrays.

# The expit function, also known as the logistic sigmoid function,
# is defined as expit(x) = 1/(1+exp(-x)). It is the inverse of the logit function.
```

Out[26]: array([[0.58471016]])

the output is greater than 0.5 yes the person is diabetic

STEP 4 PREDICTION USING MANY FEATURES

In [27]:
```
feat = df[['Glucose','BMI','Age']]


for train_index,test_index in sss.split(feat,labels):
    x_train_ , x_test_ = feat.iloc[train_index],feat.iloc[test_index]
    y_train_ , y_test_ = labels.iloc[train_index],labels.iloc[test_index]
```

In [28]:
```
x_train_.shape
```

Out[28]: (537, 3)

In [29]:
```
x_test_.shape
```

Out[29]:
```
(231, 3)
```

In [30]:
```
logreg_ = LogisticRegression()

logreg_.fit(x_train_,y_train_)
y_pred_ = logreg_.predict(x_test_)

accuracy_score(y_test_,y_pred_)
```

```
C:\Users\SURUTHI S\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a
1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[30]:
```
0.7792207792207793
```

In [31]:
```
glucose =  150
bmi = 30
age = 40

w=[glucose,bmi,age]
```

In [32]:
```
logreg_.coef_
```

Out[32]:
```
array([[0.03884178, 0.08275889, 0.02420389]])
```

In [33]:
```
logreg_.intercept_
```

Out[33]:
```
array([-8.99044306])
```

In [34]:
```
lrf_ = np.dot(logreg_.coef_,w) + logreg_.intercept_
```

In [35]:
```
lrf_
```

Out[35]:
```
array([0.28674638])
```

In [36]:
```
from scipy.special import expit

expit(lrf_)
```

Out[36]:
```
array([0.57119941])
```

the output is greater than 0.5 yes the person is diabetic

In [37]:
```
logreg_.predict_proba([w])

# the ouctome as 1 class having large probability
```

```
C:\Users\SURUTHI S\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LogisticRegressi
on was fitted with feature names
  warnings.warn(
```

Out[37]:
```
array([[0.42880059, 0.57119941]])
```

STEP 5 - BUILD LOR WITH ALL FEATURES

In [38]:
```
allFeatures = df.drop('Outcome',axis=1)

```

In [39]:
```
sss = StratifiedShuffleSplit(n_splits = 4 ,test_size = 0.30,random_state =42)

for train_index,test_index in sss.split(allFeatures,labels):
    x_train_all ,x_test_all = allFeatures.iloc[train_index],allFeatures.iloc[test_index]
    y_train_all , y_test_all = labels.iloc[train_index],labels.iloc[test_index]

```

In [40]:
```
x_train_all.shape
```

Out[40]:
```
(537, 8)
```

In [41]:
```
x_test_all.shape
```

Out[41]:
```
(231, 8)
```

In [42]:
```
logreg_all  = LogisticRegression()
logreg_all.fit(x_train_all,y_train_all)
y_pred_all = logreg_all.predict(x_test_all)

accuracy_score(y_test_all,y_pred_all)
```

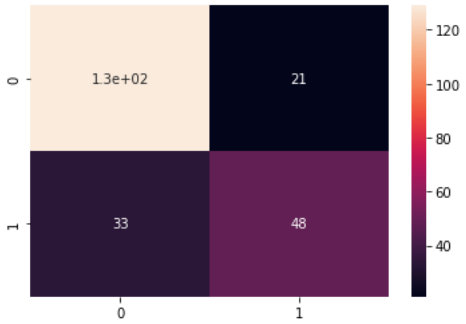
```
C:\Users\SURUTHI S\anaconda3\lib\site-packages\sklearn\utils\validation.py:993: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\SURUTHI S\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[42]:
```
0.7662337662337663
```

In [43]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test_all,y_pred_all)
```

In [44]:
```
sns.heatmap(cm,annot=True)
```

Out[44]:
```
<AxesSubplot:>
```



In [45]:
```
from sklearn.metrics import roc_auc_score

y_pred_proba = logreg_all.predict_proba(x_test_all)[:,1]
auc = roc_auc_score(y_test_all,y_pred_proba)
auc
```

Out[45]:
```
0.7995884773662552
```

https://quantifyinghealth.com/stepwise-selection/ (https://quantifyinghealth.com/stepwise-selection/)

STEP 6

In [46]:
```
import warnings
warnings.filterwarnings(action='ignore')
```

In [47]:
```
import mlxtend
```

In [48]:
```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
```

In [49]:
```
sfs = SFS(logreg_all,
          k_features=4,
          scoring='accuracy')
```

In [50]:
```
sfs.fit(x_train_all,y_train_all)
```

Out[50]:
```
SequentialFeatureSelector(estimator=LogisticRegression(), k_features=(4, 4),
                          scoring='accuracy')
```

In [51]:
```
sfs.k_feature_names_
```

Out[51]:
```
('Glucose', 'BloodPressure', 'BMI', 'DiabetesPedigreeFunction')
```

In [52]:
```
def get_next(i,model):
    sfs = SFS(model,k_features=i,scoring='accuracy')
    sfs.fit(x_train_all,y_train_all)
    return sfs.k_feature_names_

```

In [53]:
```
 columns = x_train_all.shape[1]
```

In [54]:
```
FSP = get_next(columns-1,logreg_all)
```

In [55]:
```
FSP
```

Out[55]:
```
('Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age')
```

In [56]:
```
a = FSP[0][1]
a
```

Out[56]:
```
'l'
```

In [57]:
```
def get_auc(cols):
    lr = LogisticRegression()
    x_train = x_train_all[cols]
    y_train = y_train_all

    x_test = x_test_all[cols]
    y_test = y_test_all

    lr.fit(x_train,y_train)
    ypred = lr.predict(x_test)

    y_pred_proba = lr.predict_proba(x_test)[:,1]

    auc = roc_auc_score(y_test_all,y_pred_proba)

    return auc


```

In [58]:
```
get_auc(['BloodPressure'])
```

Out[58]:
```
0.6496296296296296
```

In [59]:
```
get_auc(['BMI'])
```

Out[59]:
```
0.7565020576131688
```

```
http://rasbt.github.io/mlxtend/

https://www.codespeedy.com/sequential-forward-selection-with-python-and-scikit-learn/
```

In [ ]:
```

```

step:7 [plot Line graph of AUC values and select cut-off]

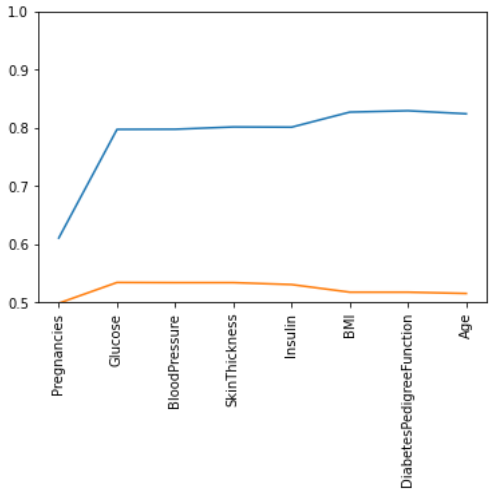
In [32]:
```
X2_train,X2_test,y2_train,y2_test=train_test_split(x_train_all,y_train_all,stratify=y_train_all,test_size=0.5,random_state=42)
```

In [33]:
```
prediction = lr2.predict_proba(x_test_all)
```

In [34]:
```
train = pd.concat([x_train_all, y_train_all], axis =1)
test = pd.concat([x_test_all, y_test_all], axis =1)
def auc_train_test (variables, target, train, test):
    X_train = train[variables]
    X_test = test[variables]
    Y_train = train[target]
    Y_test = test[target]
    lr3 = LogisticRegression()
    lr3.fit(X_train, Y_train)
    predictions_train = lr3.predict_proba(X_train)[:,1]
    predictions_test = lr3.predict_proba(X_test)[:,1]
    auc_train = roc_auc_score(Y_train, predictions_train)
    auc_test = roc_auc_score(Y_train, predictions_test)
    return (auc_train, auc_test)
auc_values_train =[]
auc_values_test =[]
variables_evaluate=[]

for v in X2.columns:
    variables_evaluate.append(v)
    auc_train, auc_test = auc_train_test(variables_evaluate, ['Outcome'],train,test)
    auc_values_train.append(auc_train)
    auc_values_test.append(auc_test)
```

...

In [35]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.array(range(0,len(auc_values_train)))
my_train = np.array(auc_values_train)
my_test = np.array(auc_values_test)
plt.xticks(x, X2.columns, rotation =90)

plt.plot(x, my_train)
plt.plot(x, my_test)
plt.ylim(0.5, 1)
plt.show()
```

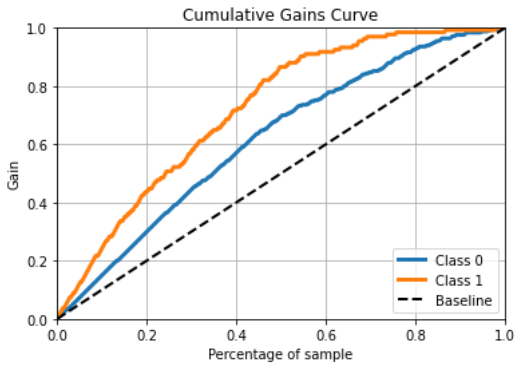


step:8 [Draw cumulative gain chart and lift chart]

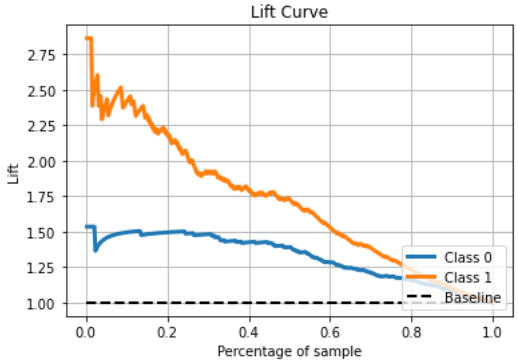
In [36]:
```
!pip install scikit-plot
from scikitplot.estimators import plot_feature_importances
from scikitplot.metrics import plot_confusion_matrix, plot_roc
```

...

In [37]:
```
import scikitplot as skplt
skplt.metrics.plot_cumulative_gain(y2_test, prediction)
plt.show()
plt.figure(figsize=(7,7))
skplt.metrics.plot_lift_curve(y2_test, prediction)
plt.show()
```




```
<Figure size 504x504 with 0 Axes>
```



In [ ]:
```
```