Libraries

In [9]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import classification_report, confusion_matrix
```

In [11]:

```
import pandas as pd
df = pd.read_csv('Diabetes.csv')

print(df)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6   
1              1       85             66             29        0  26.6   
2              8      183             64              0        0  23.3   
3              1       89             66             23       94  28.1   
4              0      137             40             35      168  43.1   
..           ...      ...            ...            ...      ...   ...   
763           10      101             76             48      180  32.9   
764            2      122             70             27        0  36.8   
765            5      121             72             23      112  26.2   
766            1      126             60              0        0  30.1   
767            1       93             70             31        0  30.4   

     DiabetesPedigreeFunction  Age  Outcome  
0                       0.627   50        1  
1                       0.351   31        0  
2                       0.672   32        1  
3                       0.167   21        0  
4                       2.288   33        1  
..                        ...  ...      ...  
763                     0.171   63        0  
764                     0.340   27        0  
765                     0.245   30        0  
766                     0.349   47        1  
767                     0.315   23        0  

[768 rows x 9 columns]
```

In [12]:

```
df.head()
```

Out[12]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

DATA OVERVIEW

In [3]:

```
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 CLEANING

In [4]:

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

Out[4]:

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

In [5]:

```
Duplicate Value
```

```
Duplicate values in df are: 0
```

In [5]:

```
print ("Duplicate values in df are:" , df.duplicated().sum())
```

```
Duplicate values in df are: 0
```

In [ ]:

```
Unique categories of Categorical Variables
```

In [6]:

```
print(df['Outcome'].unique())
```

```
[1 0]
```

Correlation

In [7]:

```
df.corr()
```

Out[7]:

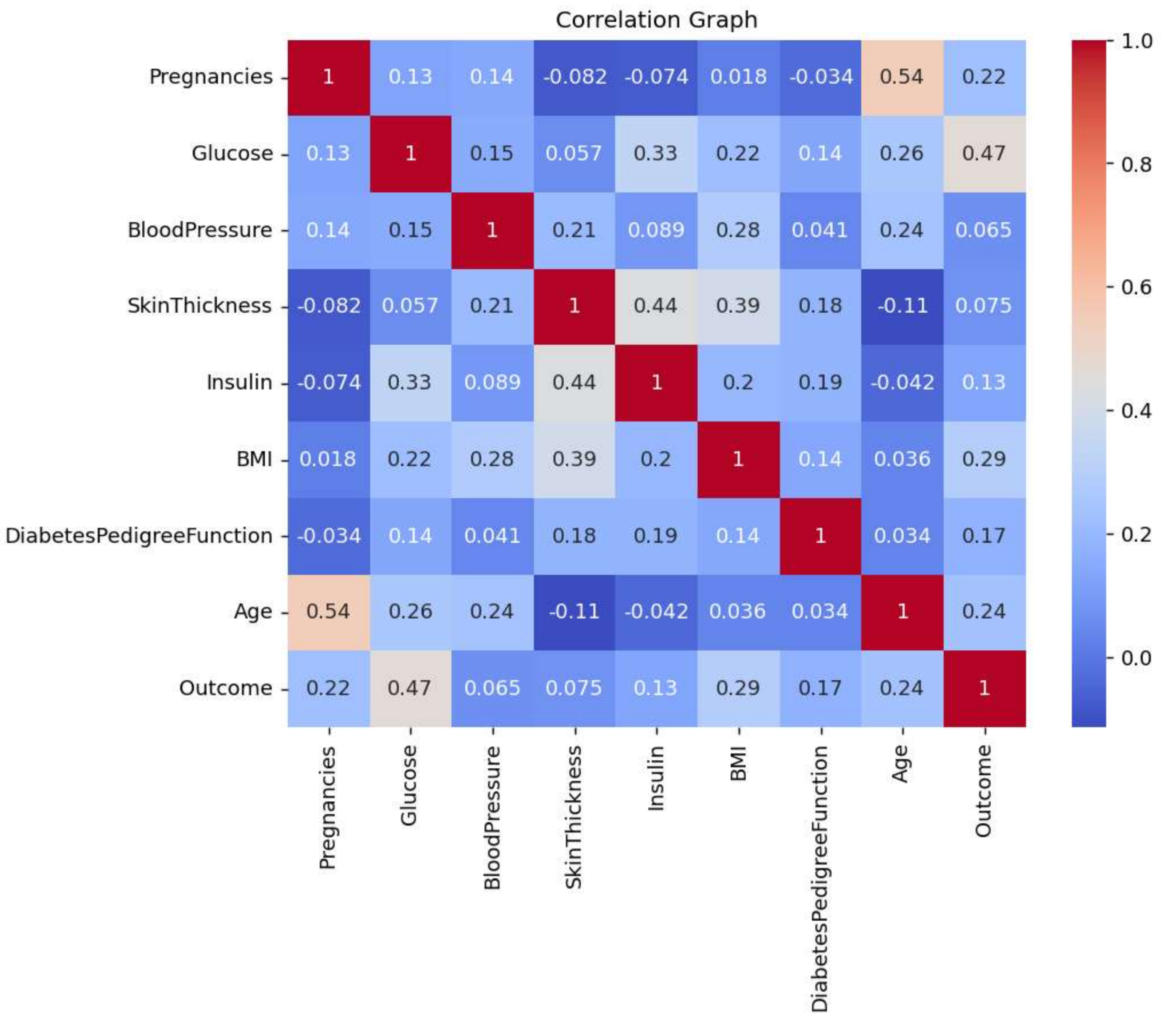
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insul
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.07353
Glucose	0.129459	1.000000	0.152590	0.057328	0.33135
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.08893
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.43678
Insulin	-0.073535	0.331357	0.088933	0.436783	1.00000
BMI	0.017683	0.221071	0.281805	0.392573	0.19785
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.18507
Age	0.544341	0.263514	0.239528	-0.113970	-0.04216
Outcome	0.221898	0.466581	0.065068	0.074752	0.13054

In [13]:

```
plt.figure (figsize = [8,6],  dpi = 130 )
plt.title ("Correlation Graph" , fontsize = 11 )
sns.heatmap (df.corr(), annot = True , cmap="coolwarm" )
```

Out[13]:

```
<Axes: title={'center': 'Correlation Graph'}>
```



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1	0.13	0.14	-0.082	-0.074	0.018	-0.034	0.54	0.22
Glucose	0.13	1	0.15	0.057	0.33	0.22	0.14	0.26	0.47
BloodPressure	0.14	0.15	1	0.21	0.089	0.28	0.041	0.24	0.065
SkinThickness	-0.082	0.057	0.21	1	0.44	0.39	0.18	-0.11	0.075
Insulin	-0.074	0.33	0.089	0.44	1	0.2	0.19	-0.042	0.13
BMI	0.018	0.22	0.28	0.39	0.2	1	0.14	0.036	0.29
DiabetesPedigreeFunction	-0.034	0.14	0.041	0.18	0.19	0.14	1	0.034	0.17
Age	0.54	0.26	0.24	-0.11	-0.042	0.036	0.034	1	0.24
Outcome	0.22	0.47	0.065	0.075	0.13	0.29	0.17	0.24	1

Split X and y into training and testing sets

In [21]:
```
X = pd.DataFrame (df , columns = ["Pregnancies" , "Glucose" , "BloodPressure" ,
                                   , "DiabetesPedigreeFunction" , "Age"]) # Feat
y = df.Outcome #

X_train , X_test , y_train , y_test = train_test_split (X , y , test_size = 0.25
```

In [22]:
```
logreg = LogisticRegression (solver = "liblinear")

logreg.fit (X_train , y_train)
y_pred = logreg.predict(X_test)
y_predicted_proba = logreg.predict_proba(X_test)
```

In [23]:
```
print ("Accuracy: " , metrics.accuracy_score (y_test , y_pred))
```

```
Accuracy:  0.8072916666666666
```

In [25]:
```
confusion_matrix (y , logreg.predict (X))
```

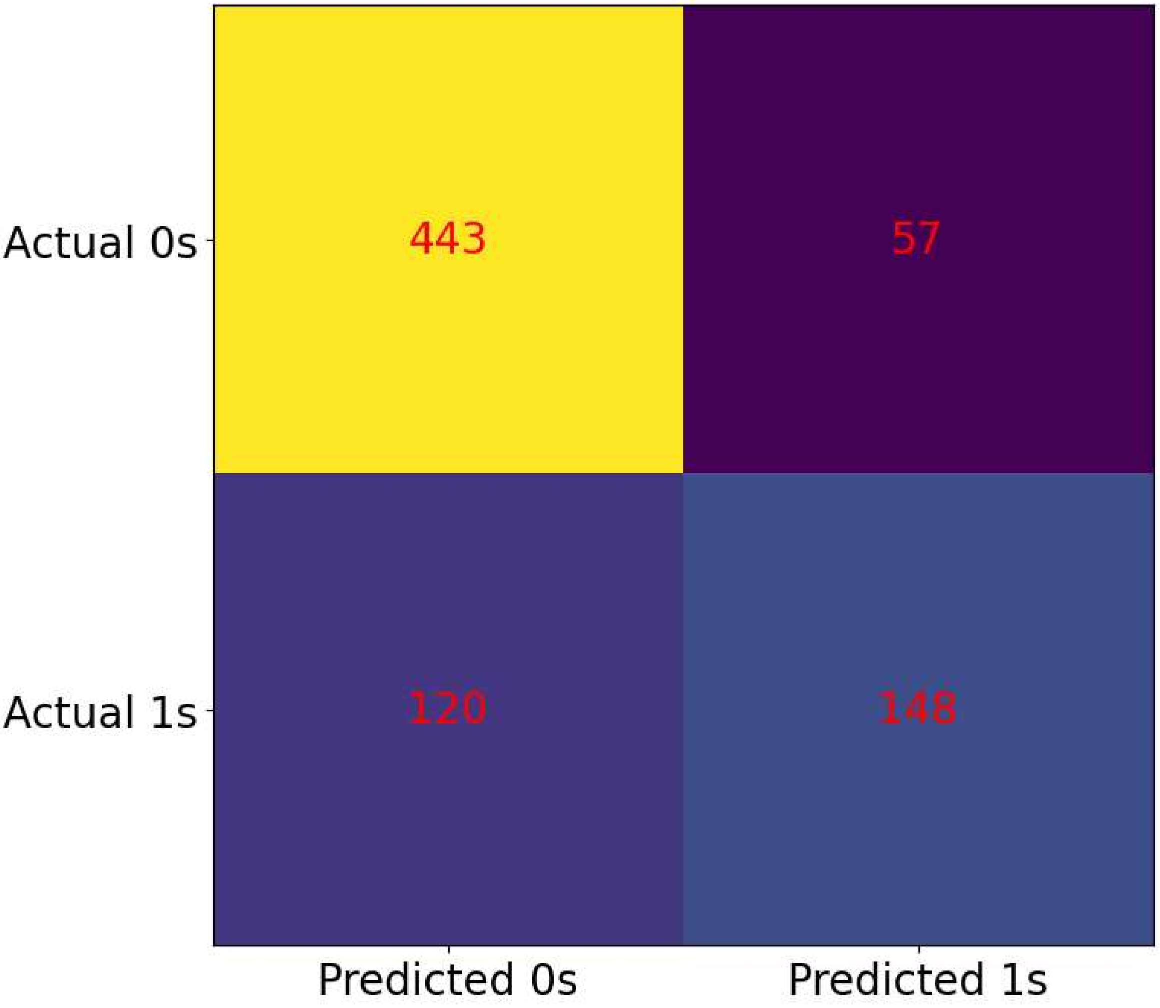
Out[25]:
```
array([[443,  57],
       [120, 148]], dtype=int64)
```

In [26]:
```
cm = confusion_matrix (y , logreg.predict(X))

fig , ax = plt.subplots (figsize = (8,8))
ax.imshow (cm)
ax.grid (False)
ax.xaxis.set (ticks = (0 , 1) , ticklabels = ("Predicted 0s" , "Predicted 1s"))
ax.yaxis.set (ticks = (0 , 1) , ticklabels = ("Actual 0s" , "Actual 1s"))

ax.tick_params(axis='both', which='major', labelsize=20)
#ax.tick_params(axis='both', which='minor', labelsize=20)
#plt.xticks(fontsize=14, rotation=90)

ax.set_ylim (1.5 , -0.5)
for i in range (2):
    for j in range (2):
        ax.text (j , i , cm[i,j] , ha = "center" , va ="center" , color ="red",
plt.show()
```



In [27]:
```
print (classification_report (y , logreg.predict (X)))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.83       500
           1       0.72      0.55      0.63       268

    accuracy                           0.77       768
   macro avg       0.75      0.72      0.73       768
weighted avg       0.76      0.77      0.76       768
```

In [ ]: