```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import classification_report, accuracy_score,f1_score,precision_score,recall_score,confusion_matrix
from sklearn.ensemble import RandomForestClassifier
```

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/DATA PROJECT/DIABETES(1).csv/diabetes(1).csv", encoding="unicode_escape")
```

```
df.head()
```

	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

```
df.tail()
```

	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
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

```
df.shape
```

```
(768, 9)
```

```
df.info()
```

```
<class pandas.core.frame.DataFrame>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Pregnancies               768 non-null  int64
 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: 54.0 KB
```

```
df.duplicated().sum()
```

```
0
```

```
df.describe()
```

	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

```
sns.heatmap(df.corr())
```

```
<Axes>
```



```
df.columns
```

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

```
cols=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']
df[cols].hist()
```

```
array([[<Axes>, <Axes>, <Axes>],
       [<Axes>, <Axes>, <Axes>],
       [<Axes>, <Axes>, <Axes>]])
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.294118	0.743719	0.590164	0.353535	0.000000	0.500745	0.234415	0.483333	1
1	0.058824	0.427136	0.526230	0.292929	0.000000	0.396423	0.116567	0.166667	0
2	0.470588	0.919598	0.524590	0.000000	0.000000	0.346939	0.253629	0.183333	1
3	0.058824	0.447236	0.526230	0.232323	0.111111	0.418778	0.036301	0.033333	0
4	0.000000	0.688442	0.327869	0.353535	0.198582	0.642325	0.943638	0.200000	1
763	0.588235	0.507538	0.622951	0.484848	0.212766	0.490313	0.039710	0.700000	0
764	0.117647	0.613065	0.573770	0.272727	0.000000	0.548435	0.071307	0.066667	0
765	0.294118	0.608040	0.590164	0.232323	0.132388	0.390462	0.051238	0.150000	0
766	0.058824	0.633166	0.491803	0.000000	0.000000	0.448584	0.073067	0.566667	1
767	0.058824	0.467085	0.573770	0.313131	0.000000	0.453055	0.065953	0.133333	0

768 rows × 9 columns

```
df.describe(include='all')
```

	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	0.226180	0.601408	0.566438	0.232415	0.115132	0.491863	0.168351	0.204013	0.348958
std	0.197459	0.160546	0.158059	0.193678	0.096231	0.119779	0.141415	0.196992	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.058824	0.497487	0.508197	0.000000	0.000000	0.415771	0.070773	0.066667	0.000000
50%	0.176471	0.587940	0.573770	0.222222	0.106383	0.491803	0.125077	0.166667	0.000000
75%	0.352941	0.703518	0.655738	0.353535	0.189716	0.561550	0.234095	0.333333	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

```
df.Outcome.value_counts
```

```
<bound_method_IndexOpsMixin.value_counts of 0    1
1    0
2    1
3    0
4    1
      ..
763  0
764  0
765  0
766  1
767  0
Name: Outcome, Length: 768, dtype: int64>
```

```
df.Outcome.value_counts()
```

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

```
sns.countplot(df.Outcome)
```

```
<Axes>
```



```
df.corr()
```

	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

```
# Exploratory Data Analysis
```

```
matplotlib.rc('xtick',labelsize=20)
matplotlib.rc('ytick',labelsize=20)
sns.countplot(df.Outcome)
```

```
<Axes>
```



```
sns.jointplot(df.Pregnancies,df.Age)
sns.heatmap(df.corr())
```

```
<Axes>
```



```
sns.lineplot(df.Pregnancies,df.Outcome)
sns.lineplot(df.Glucose,df.Outcome)
sns.lineplot(df.BloodPressure,df.Outcome)
sns.lineplot(df.SkinThickness,df.Outcome)
sns.lineplot(df.Insulin,df.Outcome)
sns.lineplot(df.BMI,df.Outcome)
sns.lineplot(df.DiabetesPedigreeFunction,df.Outcome)
sns.lineplot(df.Age,df.Outcome)
```

```
<Axes>
```



```
df.Pairplot(df.Pregnancies,df.Age)
df.jointplot(df.Pregnancies,df.Age,df.Outcome)
```

```
<Axes>
```



```
df.Pairplot(df.Pregnancies,df.Age)
df.jointplot(df.Pregnancies,df.Age,df.Outcome)
```

```
<Axes>
```



```
df
```

	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

```
# Pie chart
```

```
df.Outcome.value_counts
```

```
<bound_method_IndexOpsMixin.value_counts of 0    1
1    0
2    1
3    0
4    1
      ..
763  0
764  0
765  0
766  1
767  0
Name: Outcome, Length: 768, dtype: int64>
```

```
df.Outcome.value_counts()
```

```
Outcome
0    500
1    268
Name: count, dtype: int64
```

```
# Pie chart
```

```
plt.pie(df.Outcome.value_counts(),
        labels=['Not Diabetic','Diabetic'],
        autopct='%1.2f%%')
plt.title('Outcome in Percentage')
```

```
Text(0.5, 1.0, 'Outcome in Percentage')
```



```
# Split Data Sets
```

```
X_df.drop('Outcome',axis = 1)
y_df.Outcome
X_train,X_test,y_train,y_test = train_test_split(X_df,y_df,test_size = 0.2)
```

```
# Logistic Regression
model = LogisticRegression(max_iter=1000)
model.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
predictions = model.predict(X_test)
predictions
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       
```