Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Load the Dataset

```
dataset_url = "https://docs.google.com/spreadsheets/d/1p_WuY33JZo00wRFvtI7kEAITRHrwG0OM/export?format=csv"
df = pd.read_csv(dataset_url)
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

Analyze Dataset Information

```
print("Dataset Info:\n")
df.info()
```

```
Dataset 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
```

```
print("\nSummary Statistics:\n")
print(df.describe())
```

```
Summary Statistics:

       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
```

```
       mean    31.992578                 0.471876   33.240885    0.348958
       std      7.884160                 0.331329   11.760232    0.476951
       min      0.000000                 0.078000   21.000000    0.000000
       25%     27.300000                 0.243750   24.000000    0.000000
       50%     32.000000                 0.372500   29.000000    0.000000
       75%     36.600000                 0.626250   41.000000    1.000000
       max     67.100000                 2.420000   81.000000    1.000000
```

Check for Missing Values

```
print("\nMissing Values per Column:\n")
print(df.isnull().sum())
```

```
Missing Values per Column:

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

Count of Target Classes (0 = No Diabetes, 1 = Diabetes)

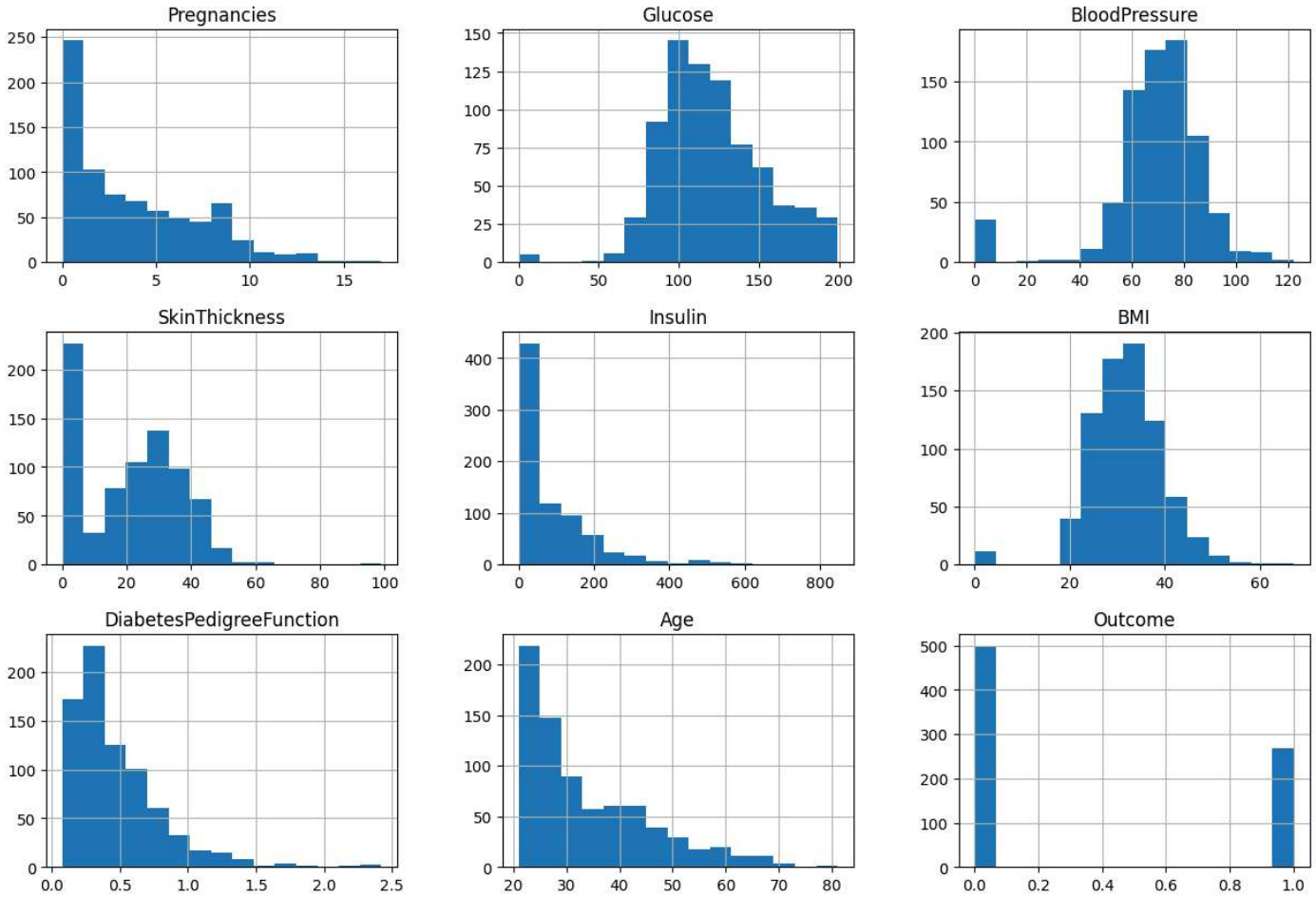
```
print("\nCount of Outcome Classes:\n")
print(df['Outcome'].value_counts())
```

```
Count of Outcome Classes:

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

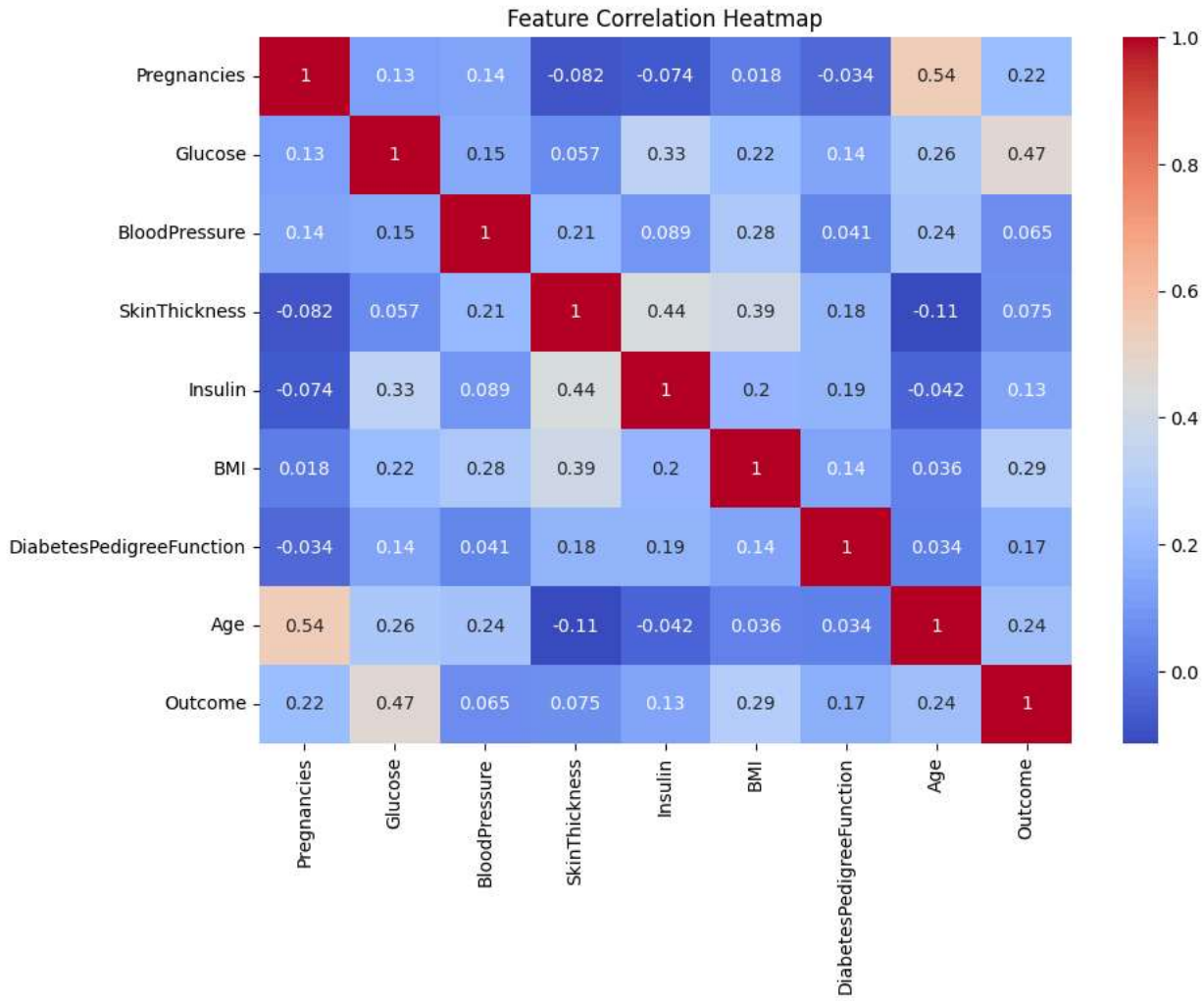
Plot Histograms for Features

```
df.hist(bins=15, figsize=(15, 10))
plt.suptitle("Feature Distributions")
plt.show()
```



Plot Correlation Heatmap

```
plt.figure(figsize=(10, 7))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```



Split Features and Target

```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
```

Split Data into Train and Test Sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Scale Features

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Train the SVM Model

```
model = SVC(kernel='rbf', probability=True)
model.fit(X_train_scaled, y_train)
```

```
SVC
SVC(probability=True)
```

Predict on Test Set

```
y_pred = model.predict(X_test_scaled)
```

Evaluate the Model

```
accuracy = accuracy_score(y_test, y_pred)
print(f"\nModel Accuracy: {accuracy * 100:.2f}%")

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Confusion Matrix Plot
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
Model Accuracy: 72.92%

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.82      0.80       123
           1       0.64      0.57      0.60        69

    accuracy                           0.73       192
   macro avg       0.71      0.69      0.70       192
weighted avg       0.72      0.73      0.73       192
```