1 Importing the Libraries

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import accuracy_score
```

2 Data Exploring

[2]:
```
# Read Dataset
data= pd.read_csv("D:\project\diabetes.csv")
```

[3]:
```
data.shape
```

[3]:
```
(768, 9)
```

[4]:
```
data.columns
```

[4]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

[5]:
```
data.head()
```

[5]:
```
   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
```

```
   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
```

[6]:
```
data.tail()
```

[6]:
```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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
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
```

[7]:
```
data.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
```

[8]:
```
#Summary Statistics of Dataset
data.describe()
```

[8]:
```
       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
```

3 Data Cleaning

[9]:

```
data.isnull().sum()
```

[9]:

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

[10]:

```
data.duplicated().sum()
```

[10]:

```
0
```

[11]:

```
#rreplacing zero values with null values
Replace_0 = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',␣
 ↪'DiabetesPedigreeFunction', 'Age']
data[Replace_0] = data[Replace_0].replace(0, np.nan)
```

[12]:

```
#removing rows if they contain more than 3 null values
rows_to_drop = data[data.isnull().sum(axis=1) > 3].index
data.drop(rows_to_drop, inplace=True)
data.reset_index(drop=True, inplace=True)
```

```
[13]: #Number of Rows in Dataset
      data.shape[0]
```

```
[13]: 761
```

```
[14]: data.groupby('Outcome').mean()
```

```
[14]:          Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
      Outcome
      0           3.297571  110.873727      70.877339      27.235457  130.287879
      1           4.846442  142.422642      75.321429      33.000000  206.846154

                     BMI  DiabetesPedigreeFunction        Age
      Outcome
      0        30.859674                  0.432261  31.285425
      1        35.406767                  0.551584  37.093633
```

```
[15]: print("Value Counts:", data['Outcome'].value_counts())
```

```
     Value Counts: 0    494
     1    267
     Name: Outcome, dtype: int64
```

```
[16]: data.nunique()
```

```
[16]: Pregnancies                  17
      Glucose                     135
      BloodPressure                46
      SkinThickness                50
      Insulin                     185
      BMI                         247
      DiabetesPedigreeFunction    515
      Age                          52
      Outcome                       2
      dtype: int64
```

```
[17]: data.isnull().sum()
```

```
[17]: Pregnancies                   0
      Glucose                       5
      BloodPressure                28
      SkinThickness               220
      Insulin                     367
      BMI                           4
      DiabetesPedigreeFunction      0
      Age                           0
      Outcome                       0
```

```
dtype: int64
```

[18]:

```
#filling null values with the mean for each column
columns_fill = ['Glucose', 'BloodPressure', 'SkinThickness','BMI', 'Insulin',␣
 ↪'Age']
mean_values = data[columns_fill].mean()
data[columns_fill] = data[columns_fill].fillna(mean_values)
```

[19]:

```
data.isnull().sum()
```

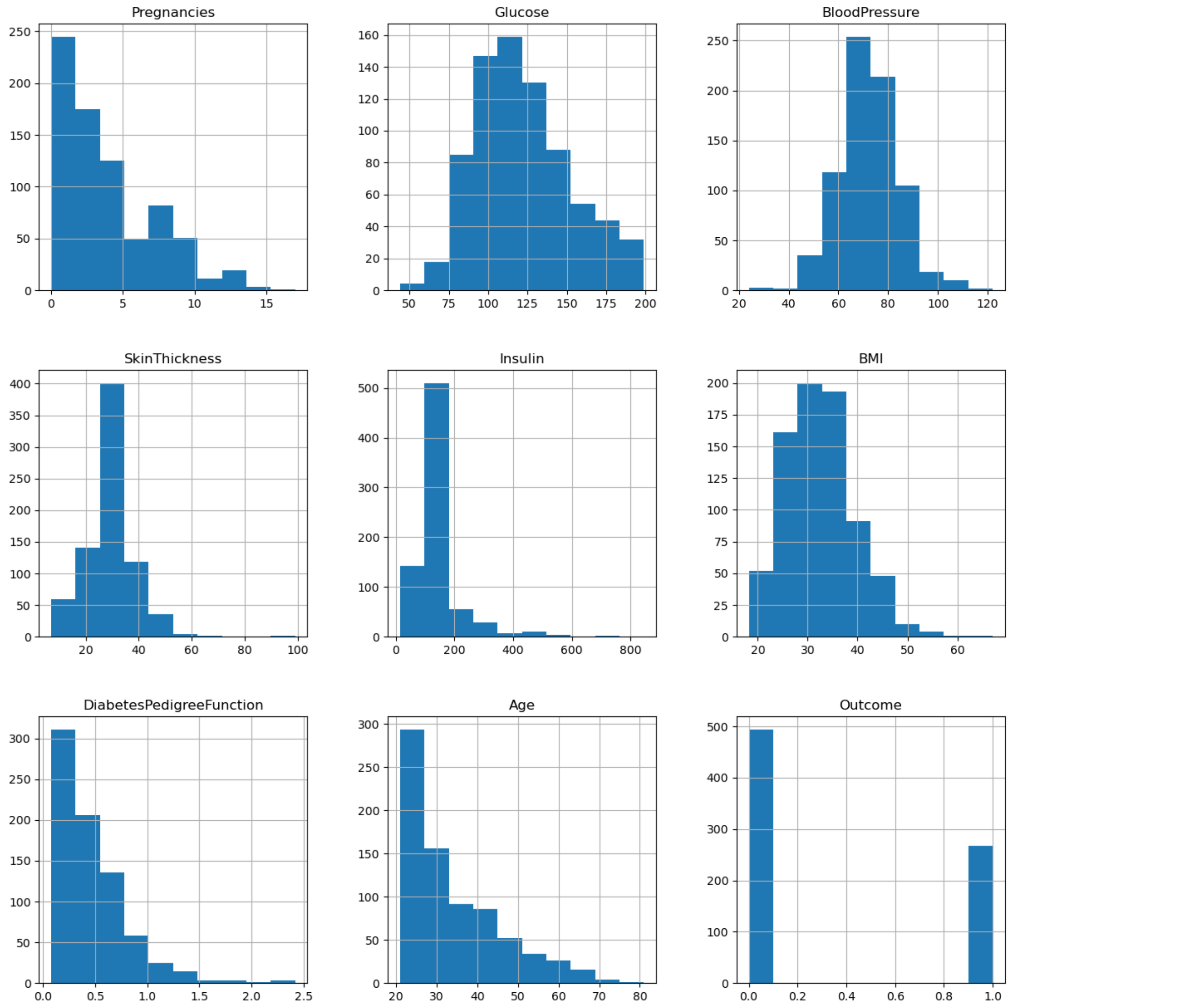
[19]:

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

4 Data Visualization

[20]:

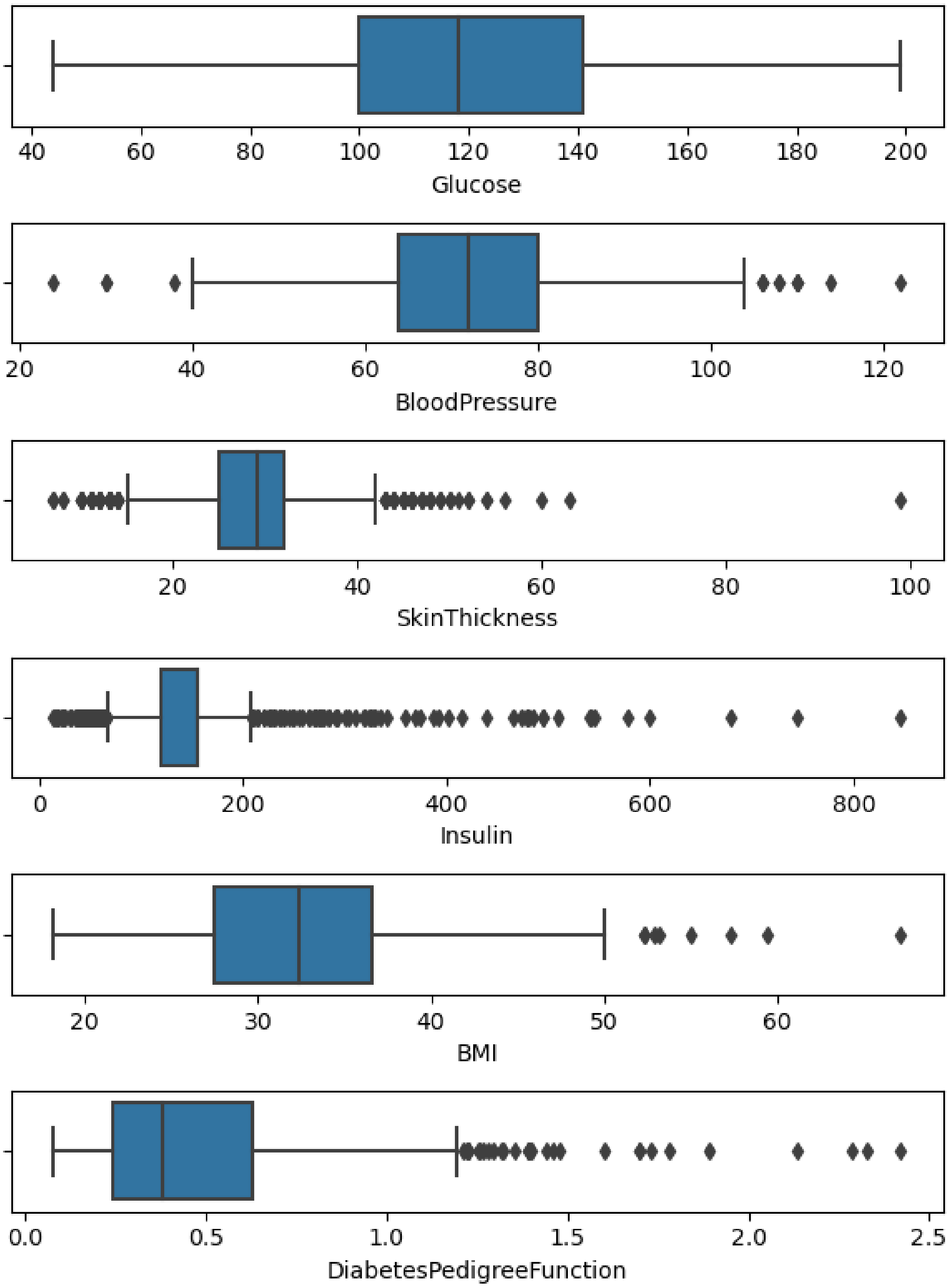
```
plots =data.hist(figsize=(15,15))
```



[21]:

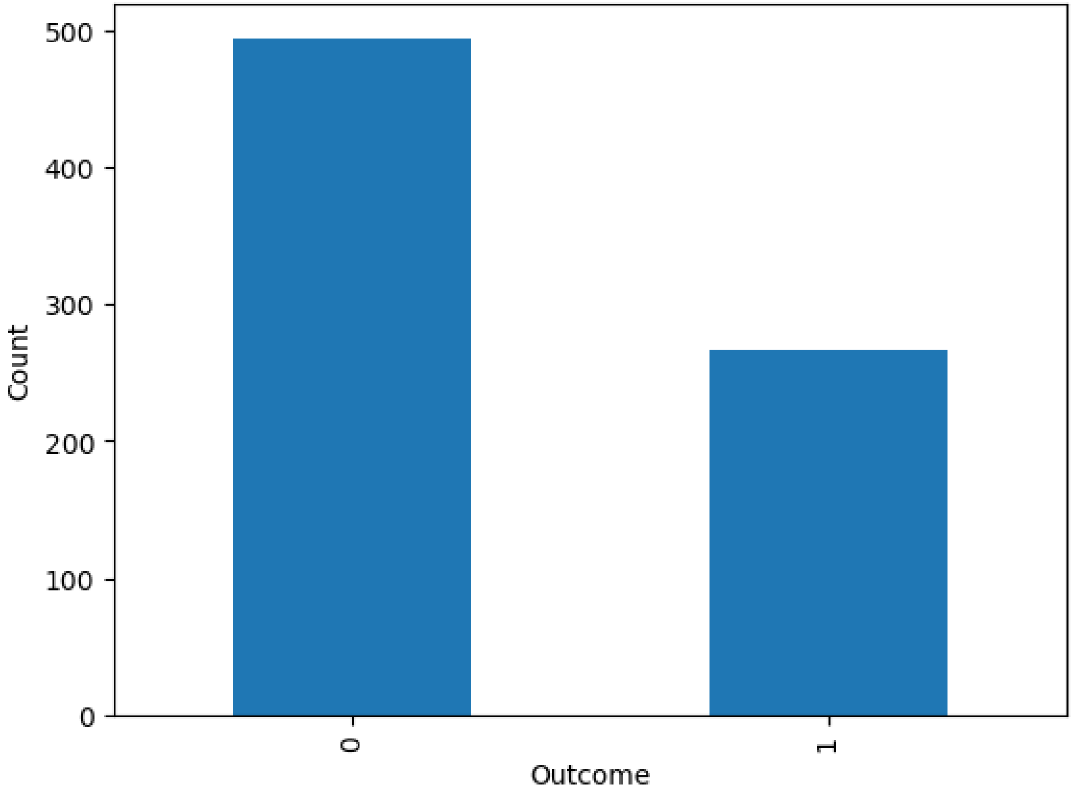
```
#Boxplots of Different Features in Diabetes Dataset
fig, axes = plt.subplots(6, 1, figsize=(6, 8))

sns.boxplot(x=data['Glucose'],ax=axes[0])
sns.boxplot(x=data['BloodPressure'],ax=axes[1])
sns.boxplot(x=data['SkinThickness'],ax=axes[2])
sns.boxplot(x=data['Insulin'],ax=axes[3])
sns.boxplot(x=data['BMI'],ax=axes[4])
sns.boxplot(x=data['DiabetesPedigreeFunction'],ax=axes[5])
plt.tight_layout()
plt.show()
```



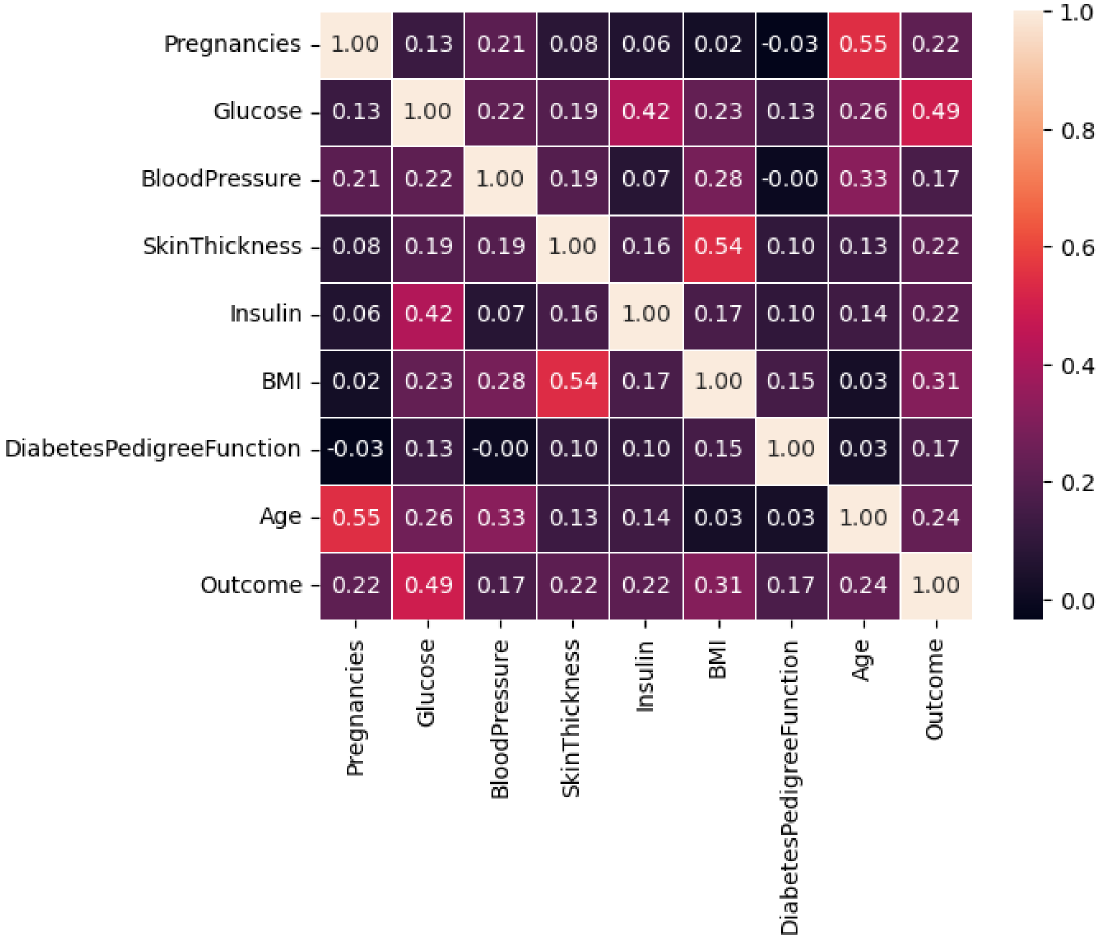
[22]:
```
#Distribution of Outcome Classes in the Dataset
plots =data.Outcome.value_counts().plot(kind="bar")
```

```
plots.set_xlabel("Outcome")
plots.set_ylabel("Count")
plt.show()
```

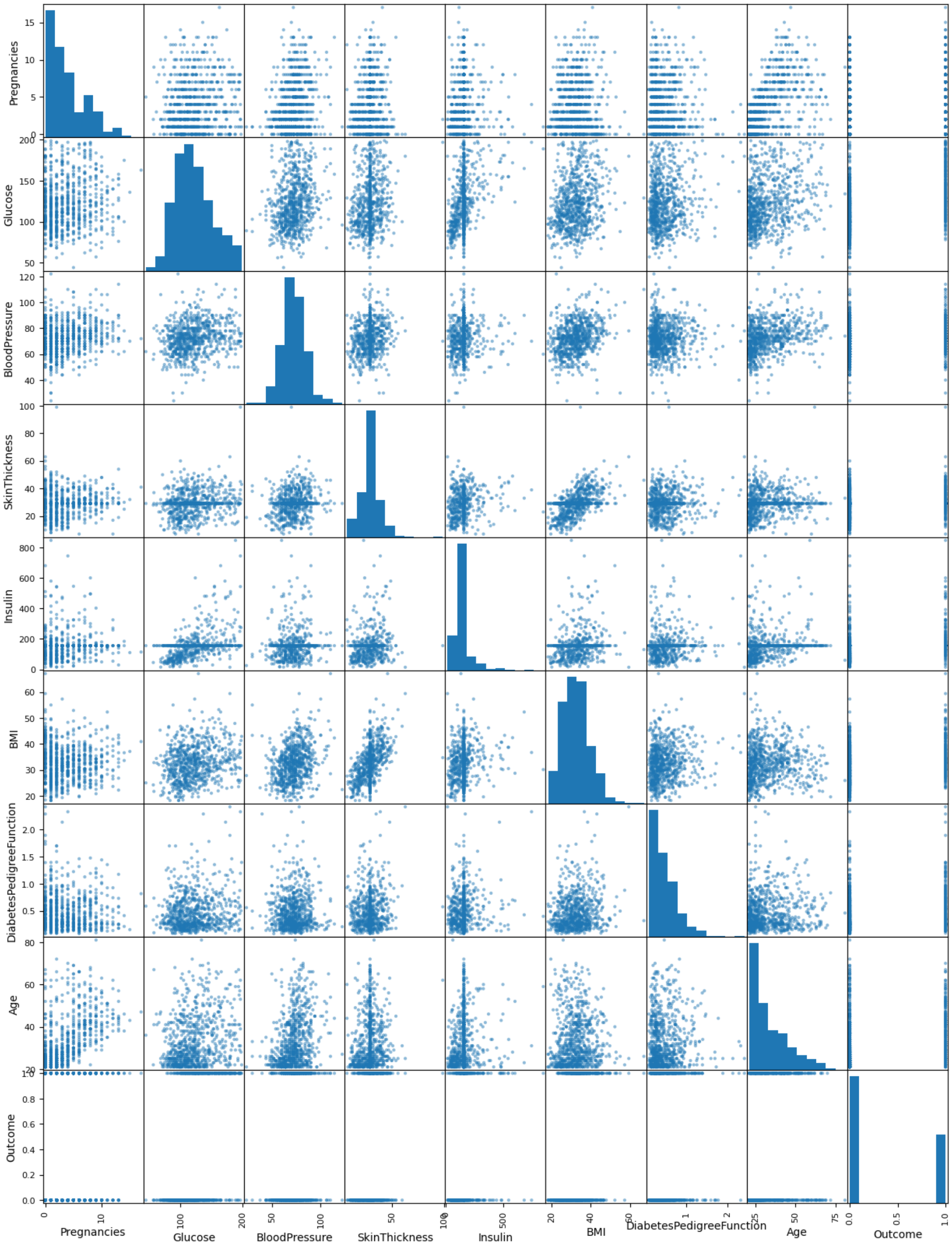


[23]:

```
#Correlation Heatmap of Dataset Features
sns.heatmap(data.corr(), annot=True, fmt=".2f", lw=0.5)
plt.show()
```



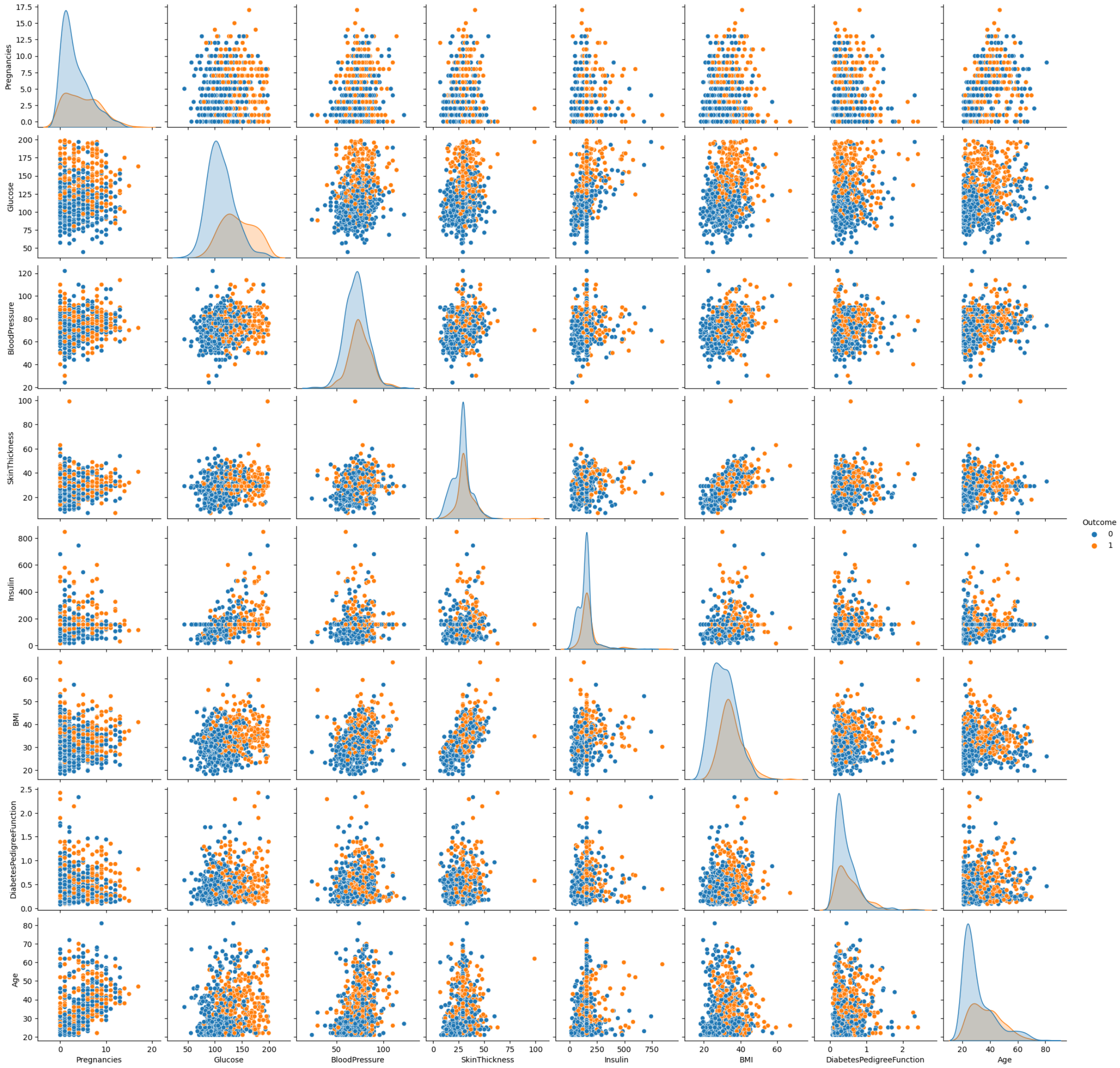
[24]:
```
plots = scatter_matrix(data , figsize=(15,20))
```



[25]:

```
#Pairplot Showing
#Healthy (Blue)
#Diabetic (Orange)
```

```
plots = sns.pairplot(data, hue='Outcome')
```



5 Splitting Data For Train and Test

[26]:
```
#separating the dependent and independent features
X = data.iloc[:, :-1].values     #features
Y = data.iloc[:, -1].values      #label
```

[27]:
```
scaler = StandardScaler()        #feature scaling
```

[28]:
```
X_Scaler = scaler.fit_transform(X)
```

[29]:
```
X_Scaler
```

[29]:
```
array([[ 0.64098111,  0.85711926, -0.03336514, …,  0.16553217,
         0.46083924,  1.41667752],
       [-0.84345779, -1.2143719 , -0.52743878, …, -0.84863705,
        -0.3711646 , -0.19735922],
       [ 1.23475666,  2.00794768, -0.69213   , …, -1.3267454 ,
         0.59649204, -0.11240992],
       …,
       [ 0.34409333, -0.03066266, -0.03336514, …, -0.90658958,
        -0.6907023 , -0.28230852],
       [-0.84345779,  0.1337414 , -1.02151243, …, -0.34155244,
        -0.37719361,  1.16182961],
       [-0.84345779, -0.9513254 , -0.19805635, …, -0.29808805,
        -0.47968684, -0.87695364]])
```

[30]:
```
X_train , X_test , Y_train , Y_test = train_test_split(X_Scaler,Y,test_size= 1/3 , random_state=0)
```

6 Modeling

[31]:
```
model = svm.SVC(kernel='linear') #support vector machine model
```

[32]:
```
model.fit(X_train,Y_train)
```

[32]:
```
SVC(kernel='linear')
```

7 Accuracy

[33]:
```
# accuracy score on the training data
pred_train = model.predict(X_train)
accuracy_score(pred_train,Y_train)
print('Model Accuracy Training Score: {0:0.2f}'.format(accuracy_score(pred_train,Y_train)*100)+ "%")
```

```
Model Accuracy Training Score: 77.12%
```

[34]:
```
# accuracy score on the test data
pred_test = model.predict(X_test)
accuracy_score(pred_test,Y_test)
print('Model Accuracy Test Score: {0:0.2f}'.format(accuracy_score(pred_test,Y_test)*100)+ "%")
```

```
Model Accuracy Test Score: 75.98%
```

8 Prediction Model

[35]:

```python
#Testing the model with input data
features_data = {
    "Pregnancies": [7],
    "Glucose": [134],
    "BloodPressure": [79],
    "SkinThickness": [0],
    "Insulin": [84],
    "BMI": [24.8],
    "DiabetesPedigreeFunction": [0.230],
    "Age": [51]
}

outcome_predict = model.predict(pd.DataFrame(features_data))
print(outcome_predict)

if outcome_predict[0] == 0:
    print("doesn't have Diabetes.")
else:
    print("have Diabetes")
```

```
[1]
have Diabetes
```