Capstone_diebetes

September 8, 2021

Project Task Week 1

Descriptive Analysis

[5]:
```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
from sklearn import metrics
```

[6]:
```
Dia_Data = pd.read_csv('/home/labsuser/Python/Datasets/Capstone_diabetes.csv')
```

[7]:
```
Dia_Data
```

[7]:
```
     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]
```

[8]:
```
#Checking the no of zeroes present in the datasets for different fields
```

[9]:
```
(Dia_Data == 0).sum()
```

[9]:
```
Pregnancies                 111
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

[10]:
```
#This shows that there are missing values for fields Glucose, Bloodpressure,␣
 ↪SkinThickness, Insulin , BMI
#as these fields cannot be zero.
```

[11]:
```
Dia_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
```

[12]:
```
#The above also shows the data type and whether the fields are having null␣
 ↪values or not.
#The above shows we have no null value in the dataset.
```

[13]:
```
Dia_Data.describe()
```

[13]:

	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

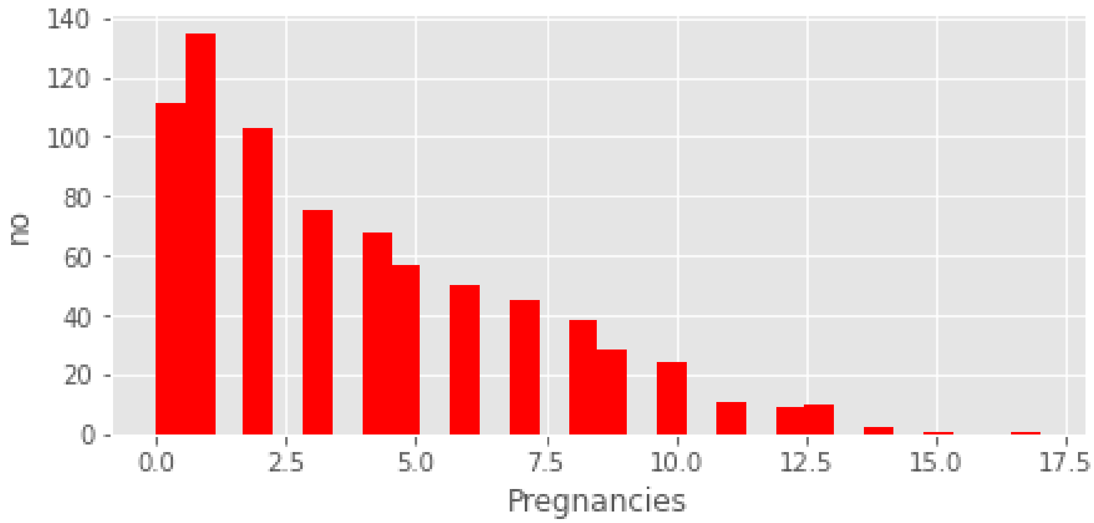
	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000
max	67.100000	2.420000	81.000000	1.000000

[14]:
```
#The above shows the various measures of central tendcies.
```

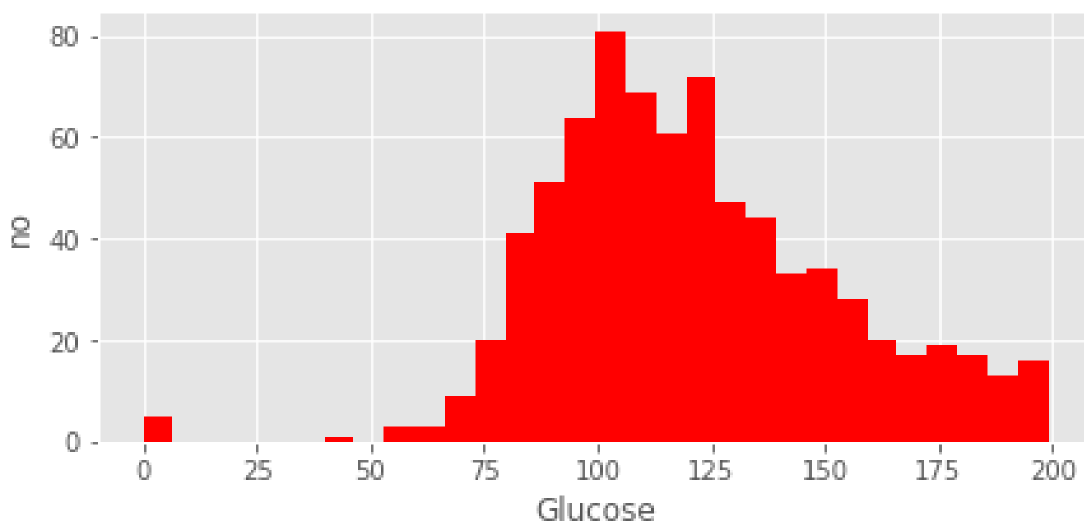
Visually Exploring the variables in the datasets

[15]:
```
import matplotlib.pyplot as plt
from matplotlib import style
for each in Dia_Data.columns:
    Col = str(each)
    print(Col)
    Data = Dia_Data[Col]
    style.use('ggplot')
    plt.figure(figsize=(7,7))
    plt.subplots_adjust(hspace=.25)
    plt.subplot(2,1,1)
    plt.hist(Data,bins= 30,color='red')
    plt.xlabel(each)
    plt.ylabel('no')
    plt.show()
```

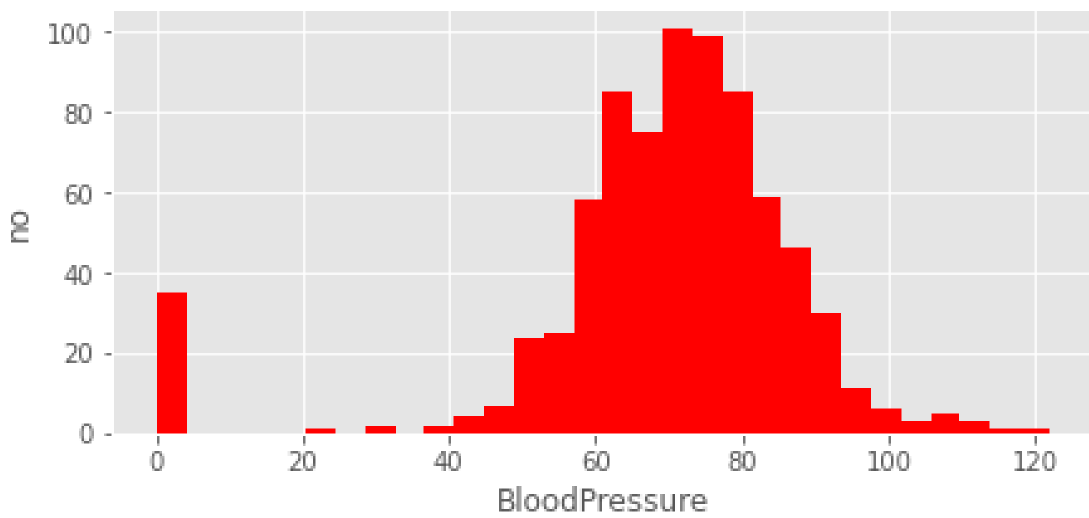
```
Pregnancies
```



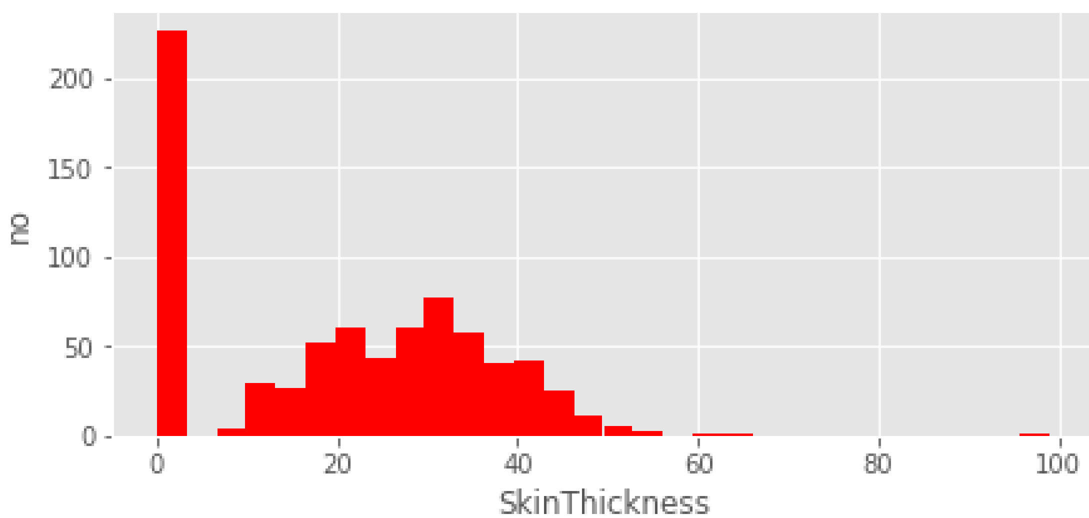
Glucose



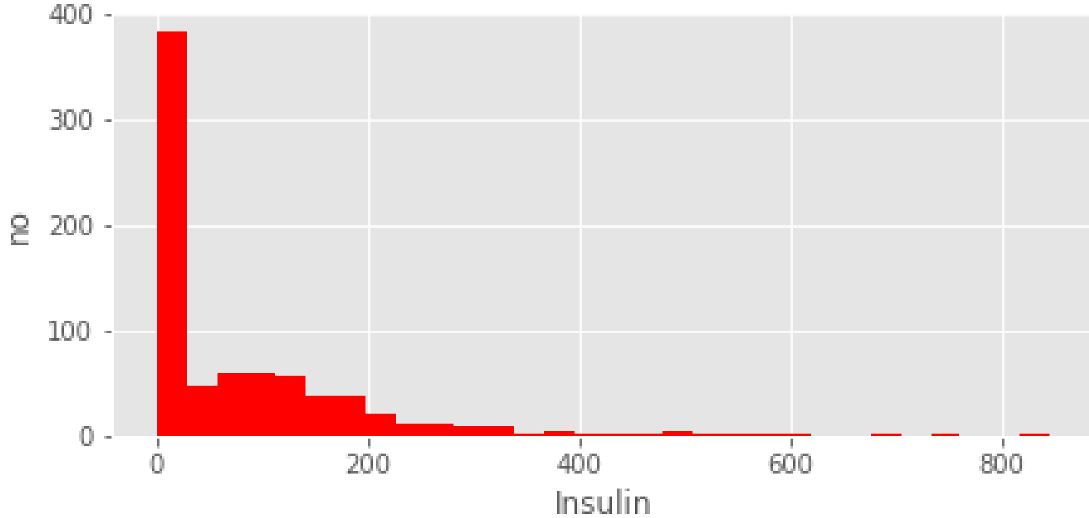
BloodPressure



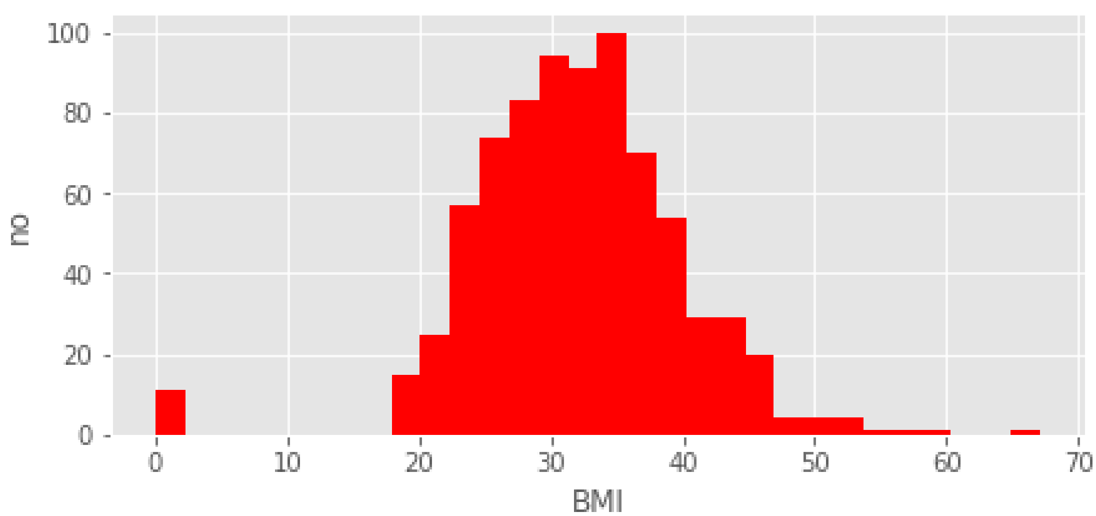
SkinThickness



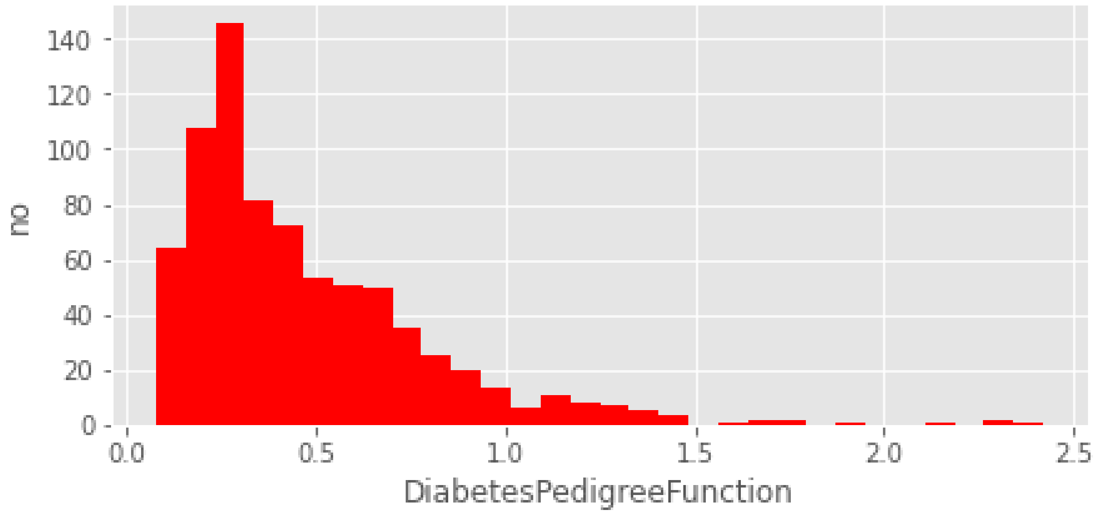
Insulin



BMI

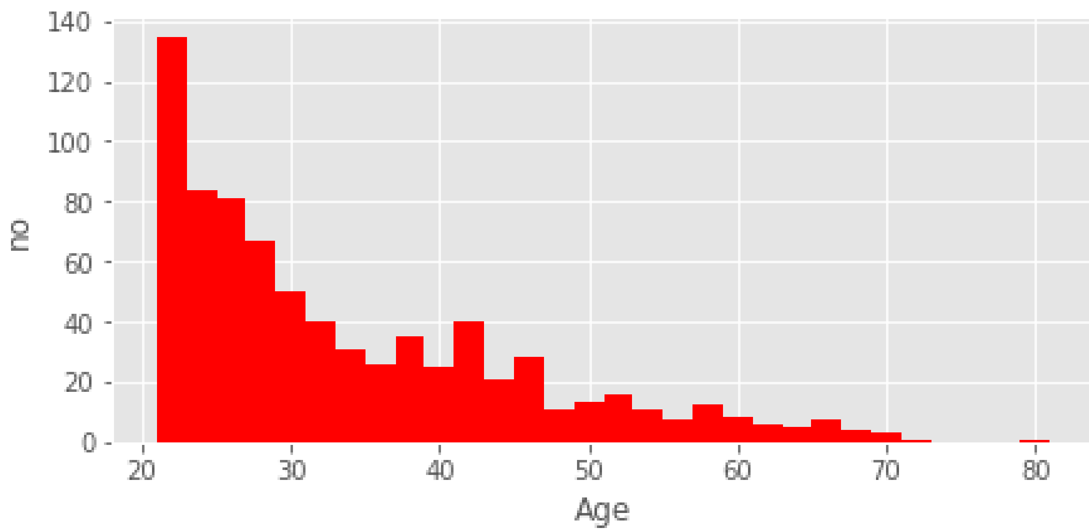


DiabetesPedigreeFunction



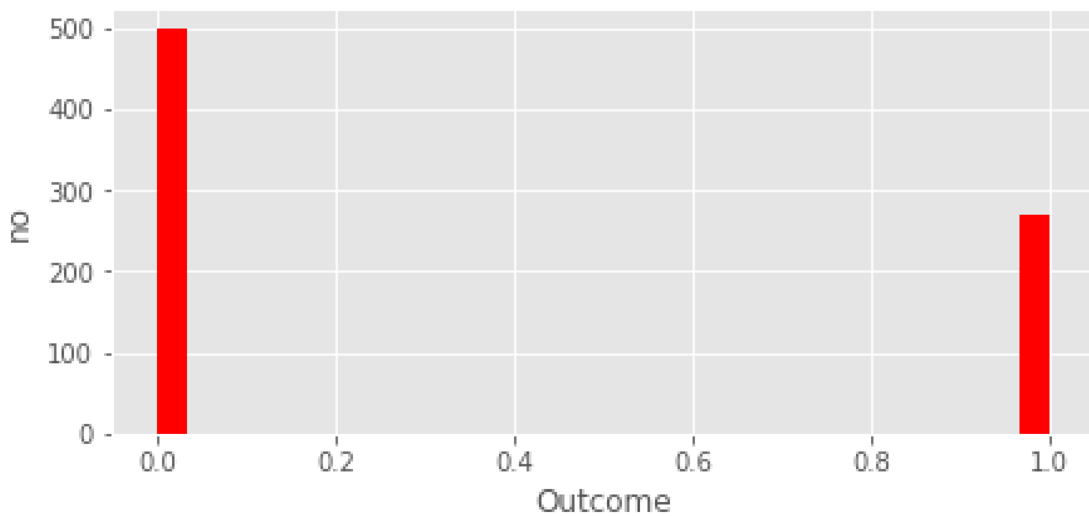
140
120
100
80
60
40
20
0
no
0.0
0.5
1.0
1.5
2.0
2.5
DiabetesPedigreeFunction

Age



140
120
100
80
60
40
20
0
no
20
30
40
50
60
70
80
Age

Outcome



[16]:
```
#Using Median Values from the above coomands describe result, and using it to␣
↪replace.
```

[17]:
```
Dia_Data['Glucose'].replace(0,117,inplace=True)
Dia_Data['BloodPressure'].replace(0,72,inplace=True)
Dia_Data['SkinThickness'].replace(0,23,inplace=True)
Dia_Data['Insulin'].replace(0,30.5,inplace=True)
Dia_Data['BMI'].replace(0,32,inplace=True)
```

[18]:
```
(Dia_Data==0).sum()
```

[18]:
```
Pregnancies                 111
Glucose                       0
BloodPressure                 0
SkinThickness                 0
Insulin                       0
BMI                           0
DiabetesPedigreeFunction      0
Age                           0
Outcome                     500
dtype: int64
```

[19]:
```
#the above shows that the zero values in the dataset have been replaced .
```

[20]:
```
#Create a count (frequency) plot describing the data types and the count of␣
↪variables.
```

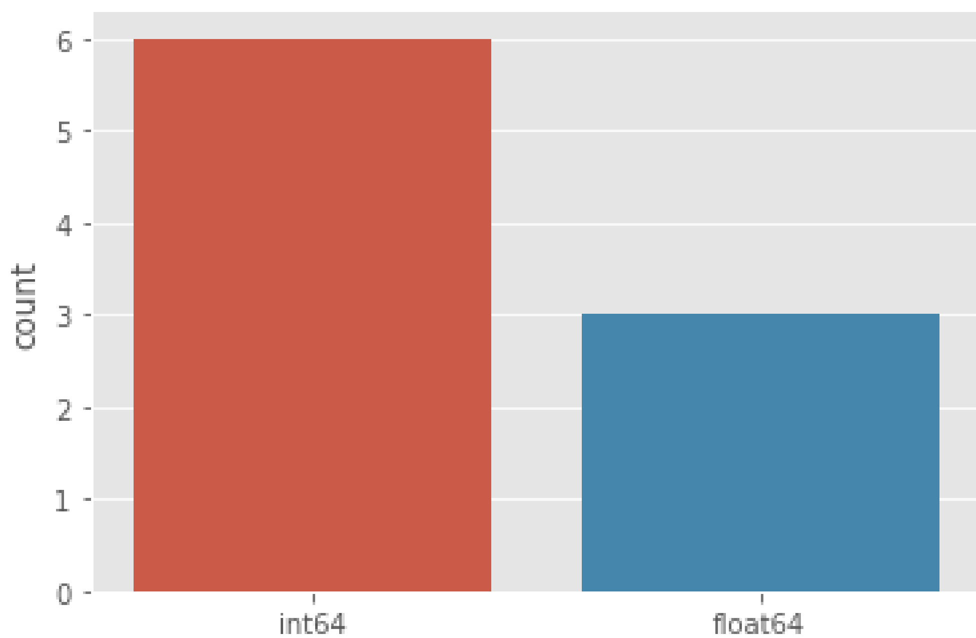
[21]:
```
Data_Type = []
for each in Dia_Data.columns:
```

```
    Col = str(each)
    Data_Type.append(str(Dia_Data[Col].dtype))

print(Data_Type)
sns.countplot(Data_Type)
```

```
['int64', 'int64', 'int64', 'int64', 'float64', 'float64', 'float64', 'int64',
'int64']
```

[21]: <AxesSubplot:ylabel='count'>

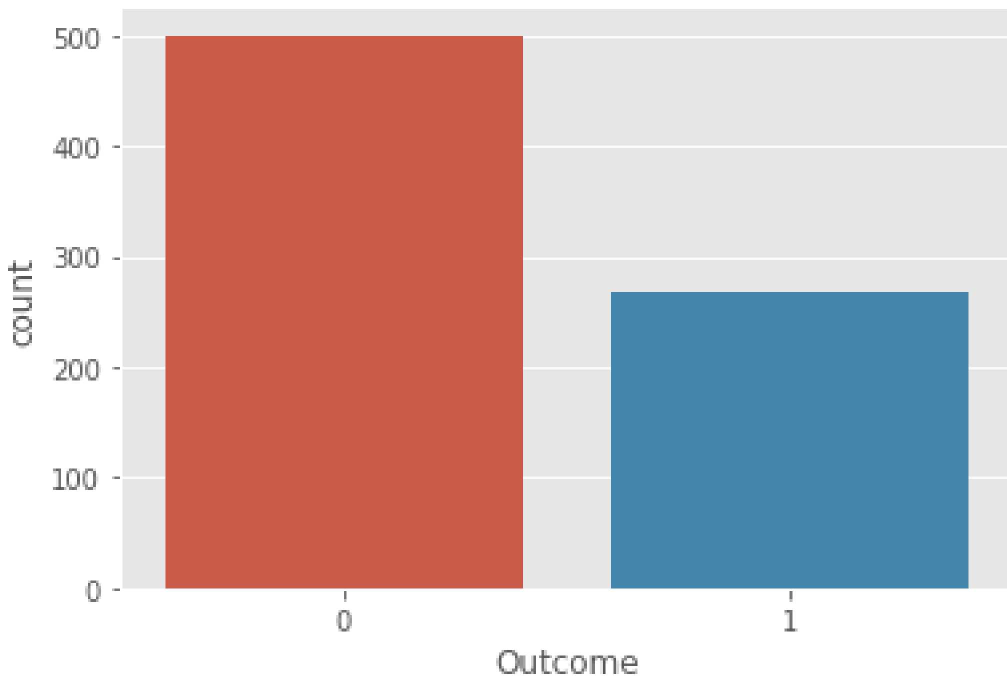


Project Task Week 2

[22]:
```
#Check the balance of the data by plotting the count of outcomes by their value.
#Describe your findings and plan future course of action.
```

[23]:
```
sns.countplot(Dia_Data.Outcome)
```

[23]: <AxesSubplot:xlabel='Outcome', ylabel='count'>

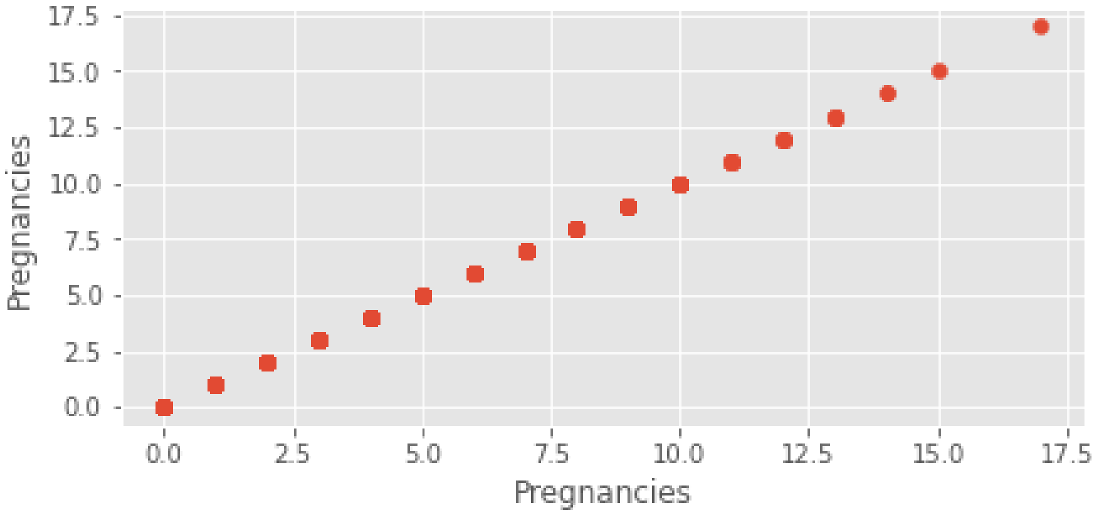


[24]:
```
# Since Both the outcomes i.e 1 and 0 are are consiserable, so the dataset is␣
↪balanced
```

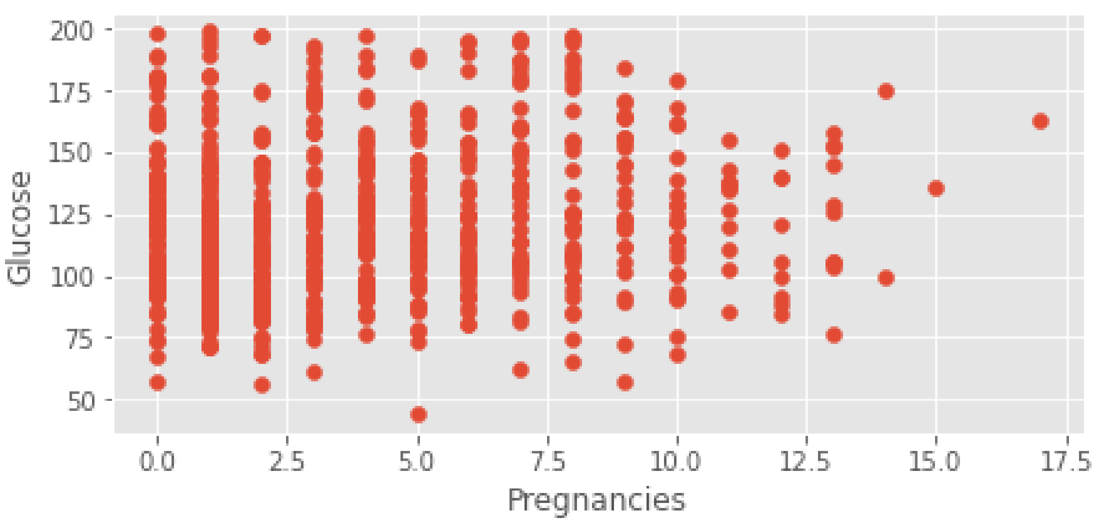
[25]:
```
#2. Create scatter charts between the pair of variables to understand the␣
↪relationships. Describe your findings.
```

[26]:
```
import matplotlib.pyplot as plt
from matplotlib import style
for each in Dia_Data.columns:
    for one in Dia_Data.columns:
        Col1 = str(each)
        Col2 = str(one)
        print(Col2,'vs',Col1)
        Data1 = Dia_Data[Col1]
        Data2 = Dia_Data[Col2]
        style.use('ggplot')
        plt.figure(figsize=(7,7))
        plt.subplots_adjust(hspace=.25)
        plt.subplot(2,1,1)
        plt.scatter(Data1,Data2)
        plt.xlabel(each)
        plt.ylabel(one)
        plt.show()
```

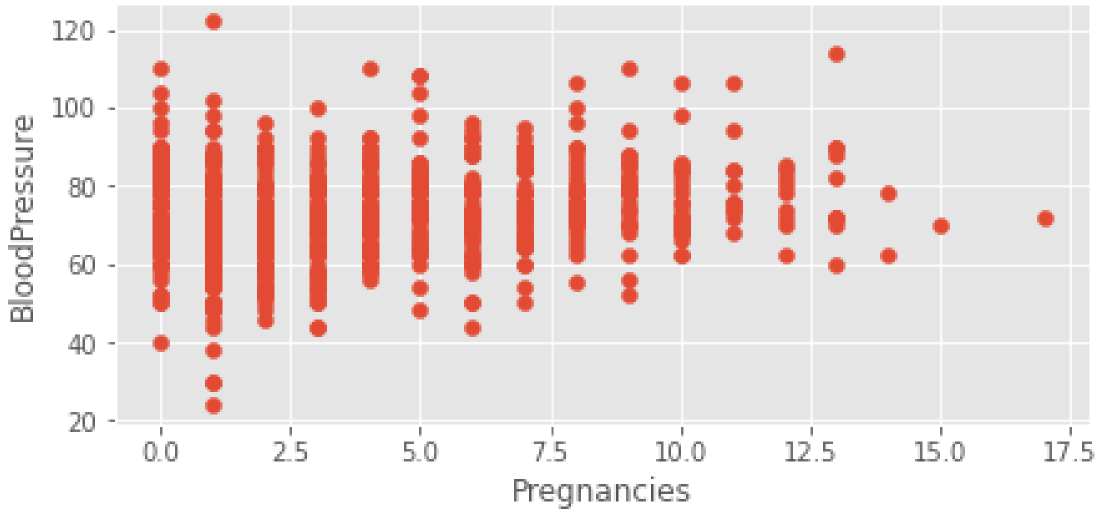
```
Pregnancies vs Pregnancies
```



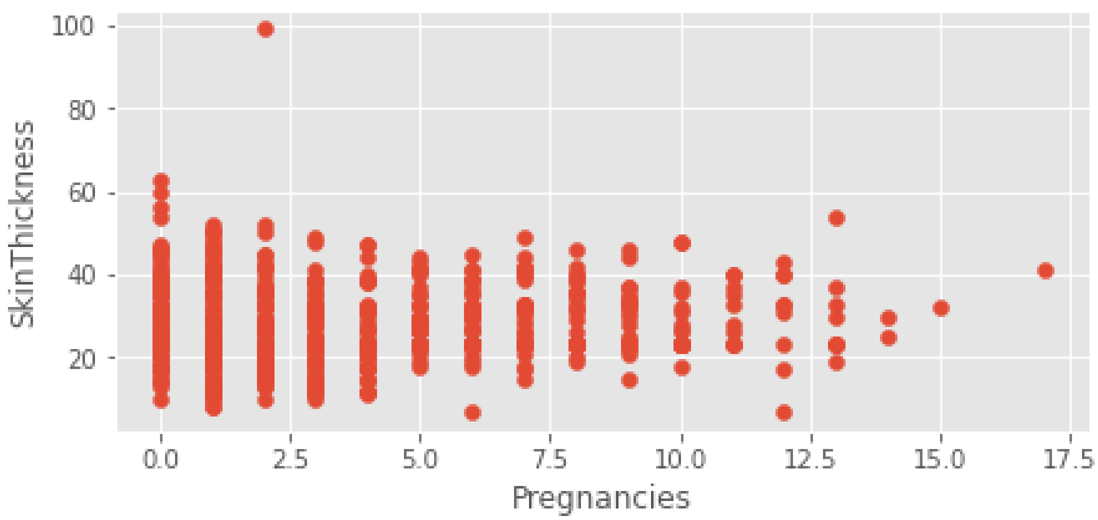
Glucose vs Pregnancies



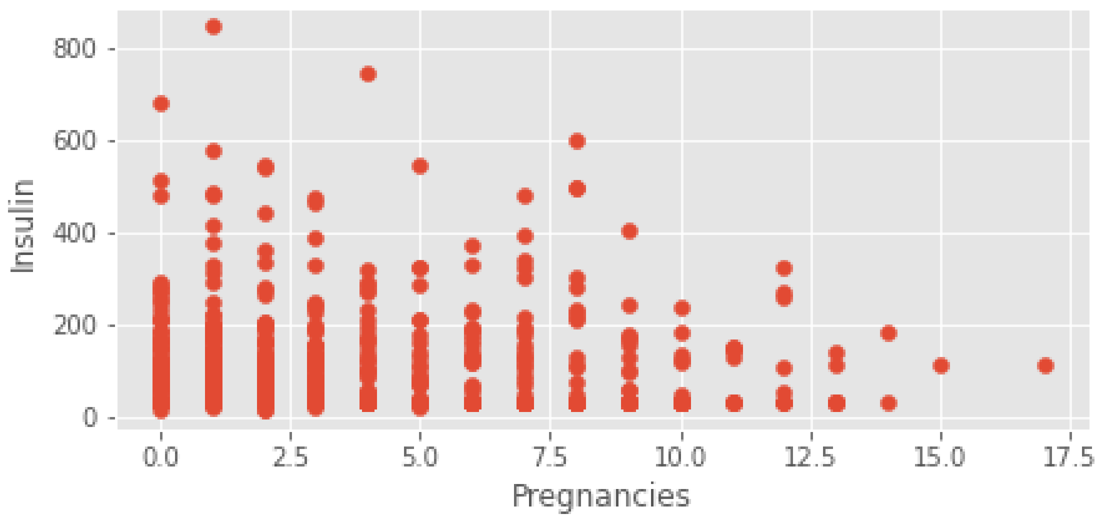
BloodPressure vs Pregnancies



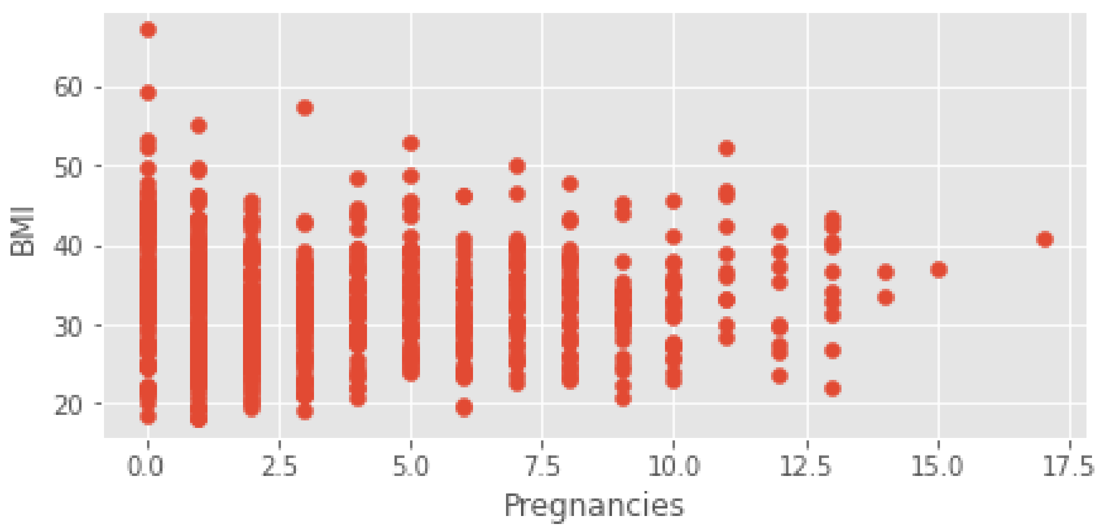
SkinThickness vs Pregnancies



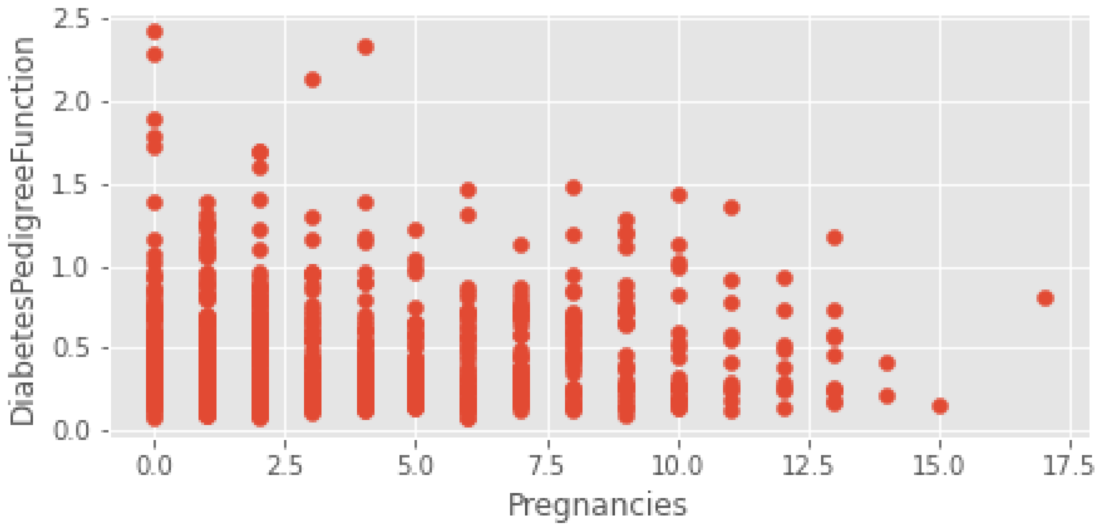
Insulin vs Pregnancies



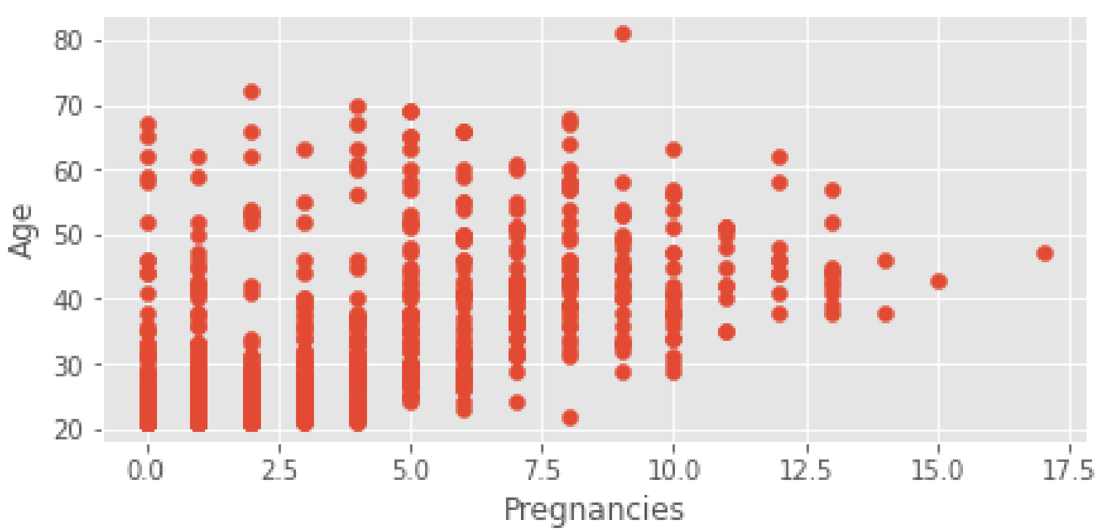
BMI vs Pregnancies



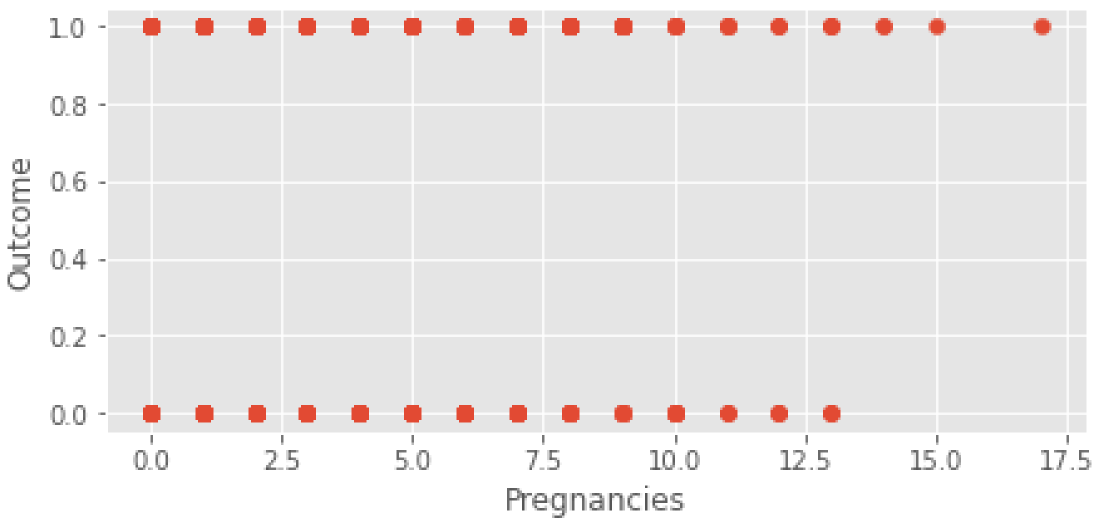
DiabetesPedigreeFunction vs Pregnancies



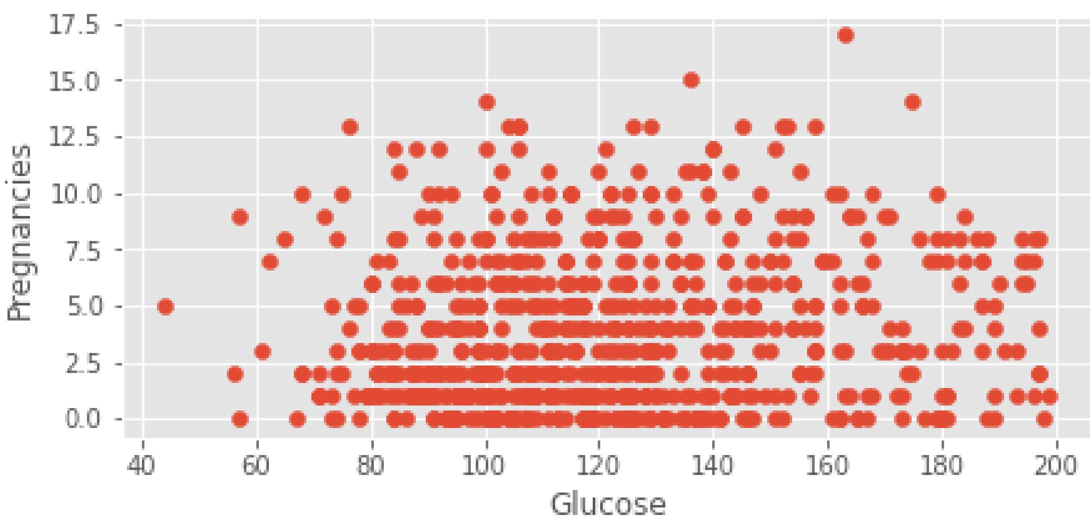
Age vs Pregnancies



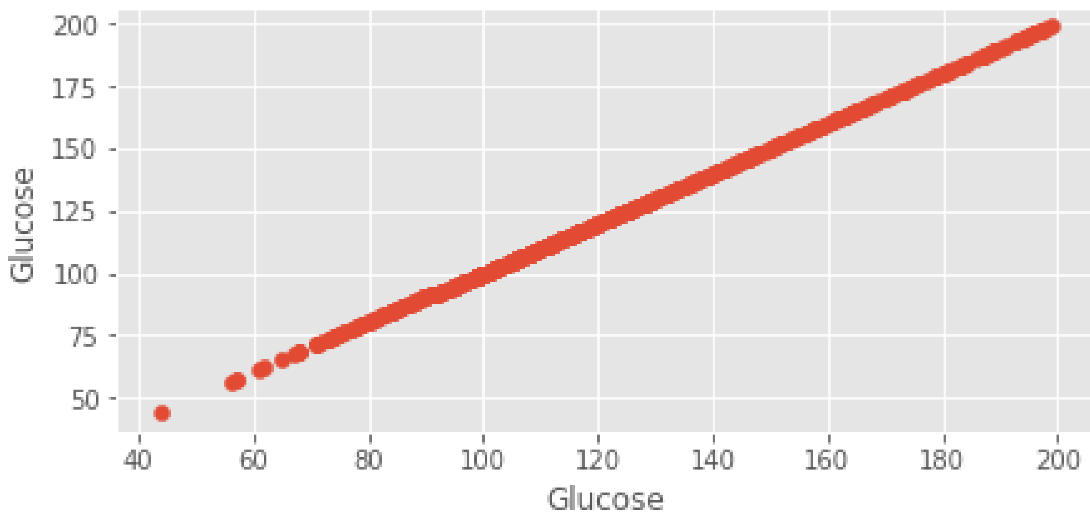
Outcome vs Pregnancies



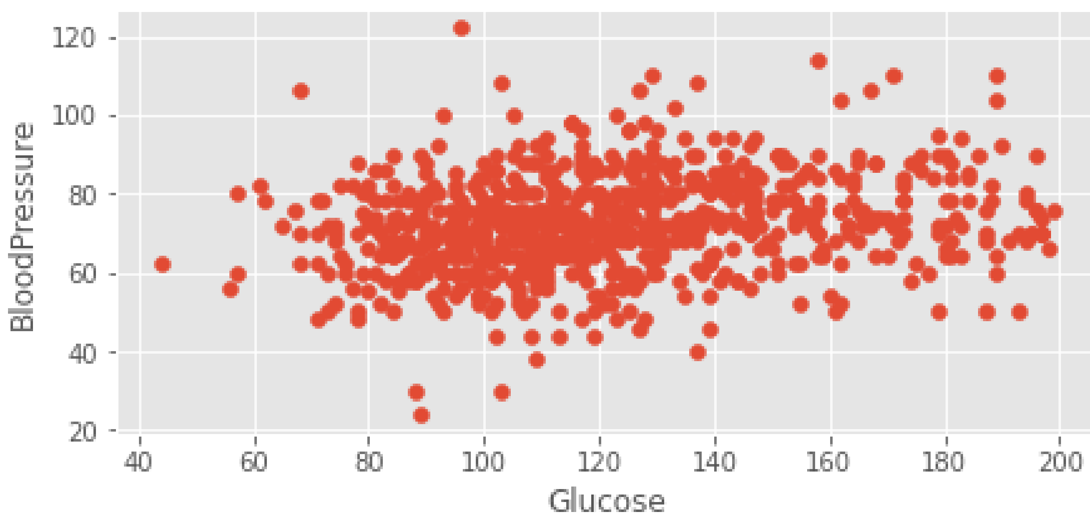

```
Pregnancies vs Glucose
```



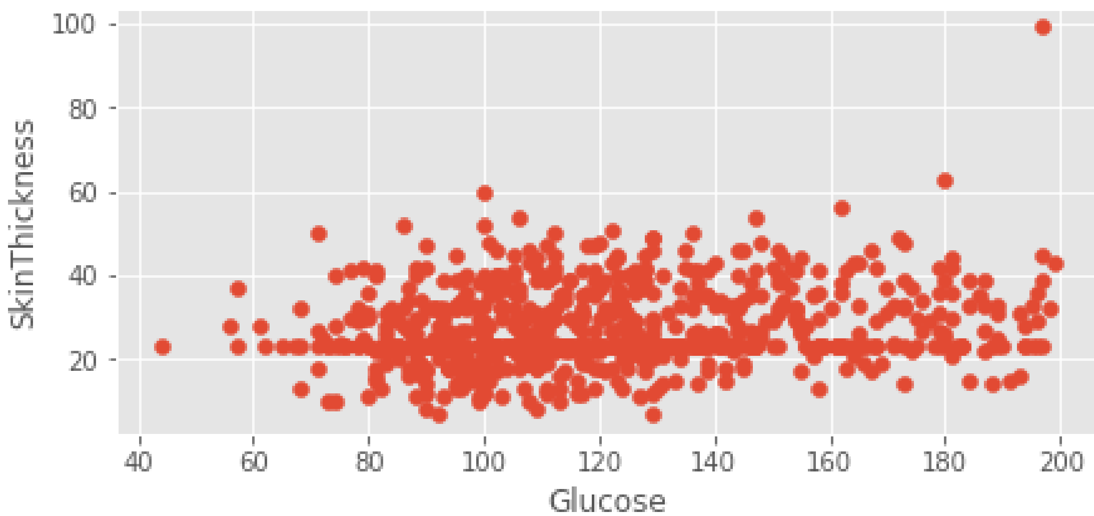

```
Glucose vs Glucose
```



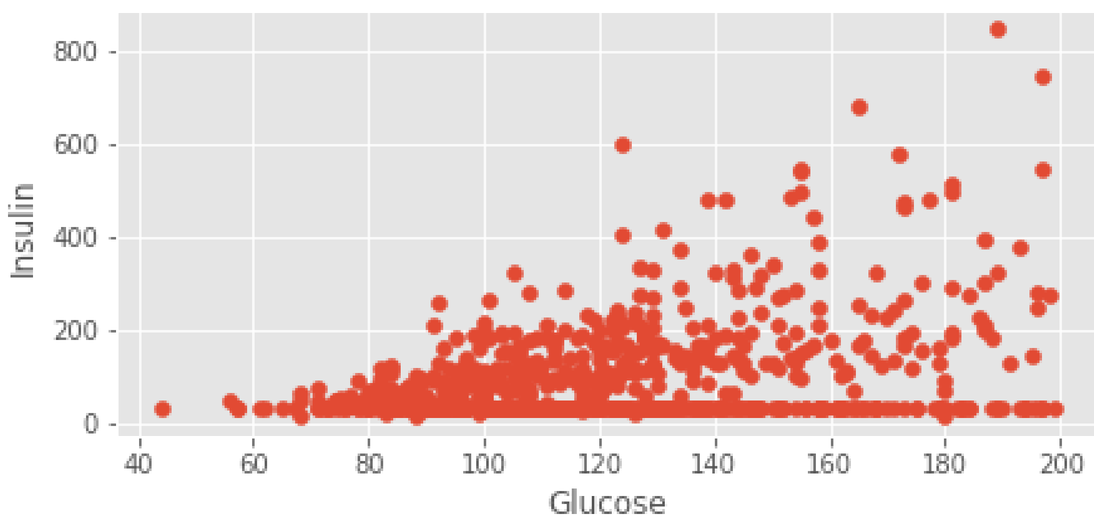
BloodPressure vs Glucose



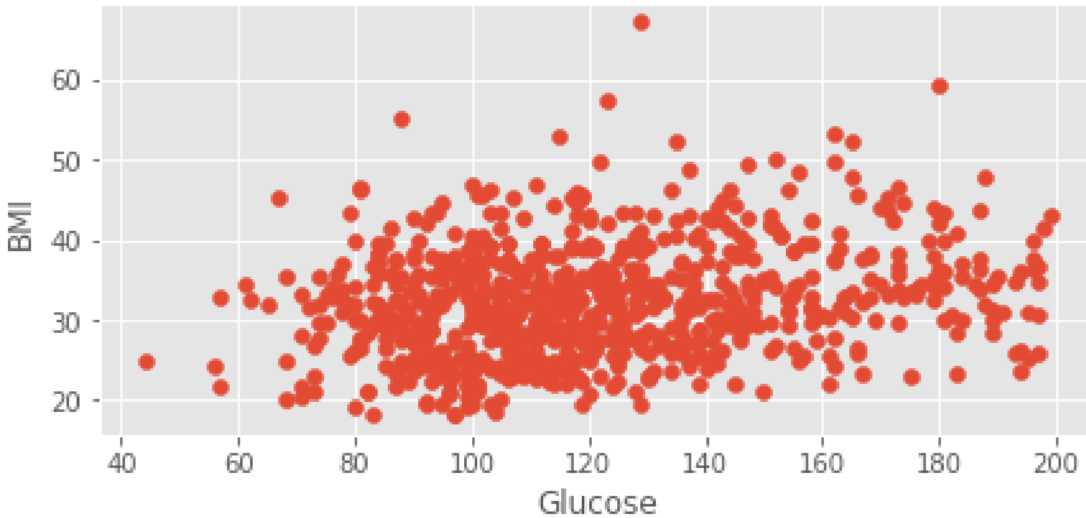
SkinThickness vs Glucose



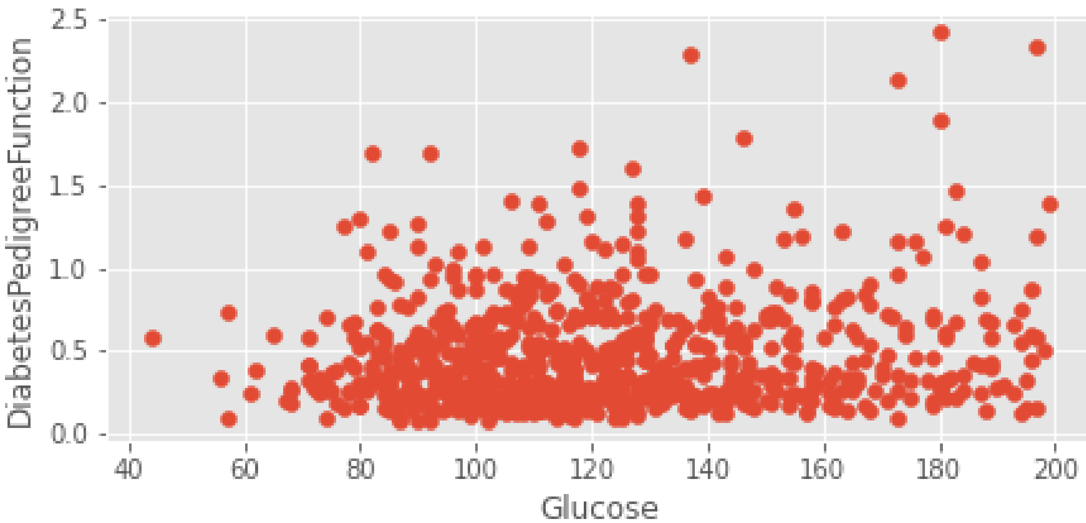
Insulin vs Glucose



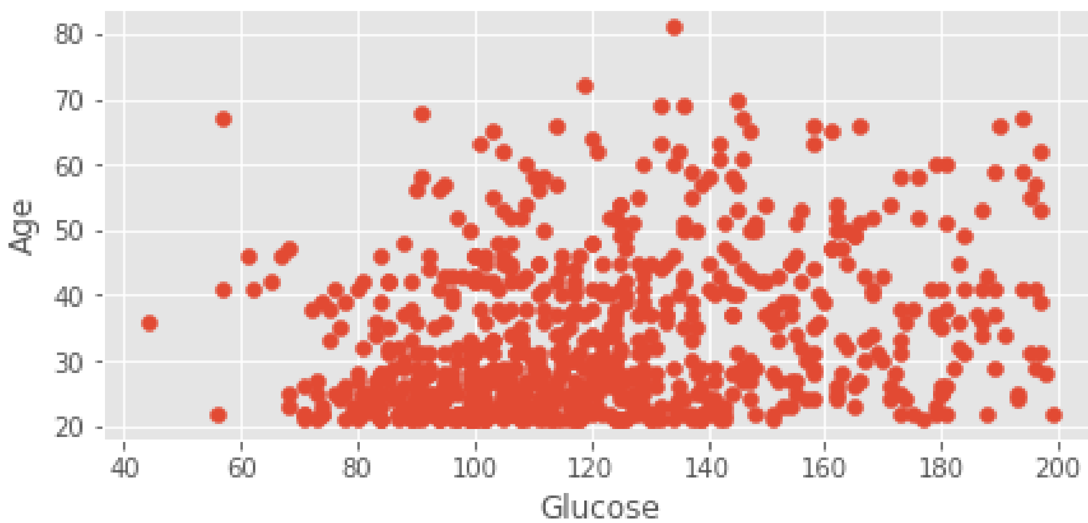
BMI vs Glucose



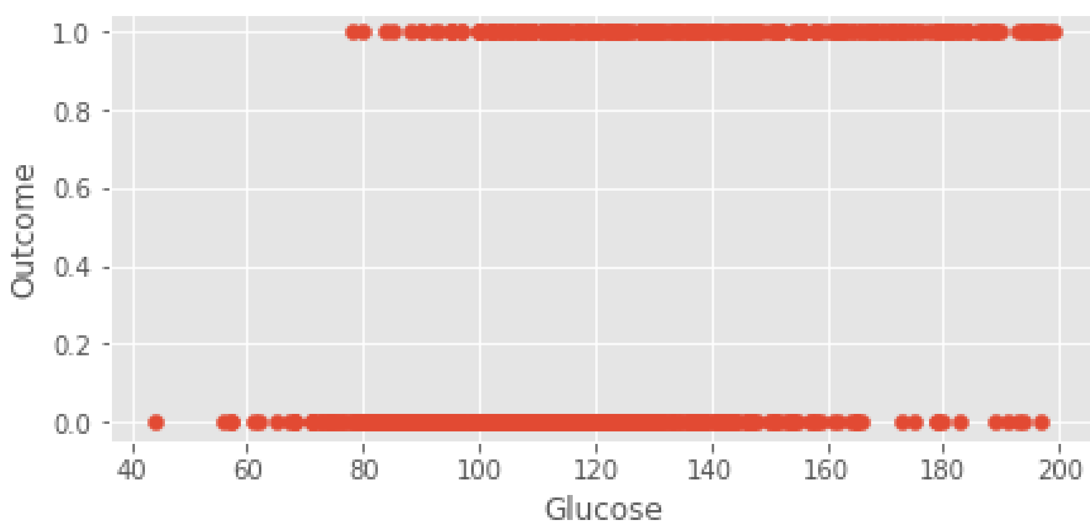
DiabetesPedigreeFunction vs Glucose



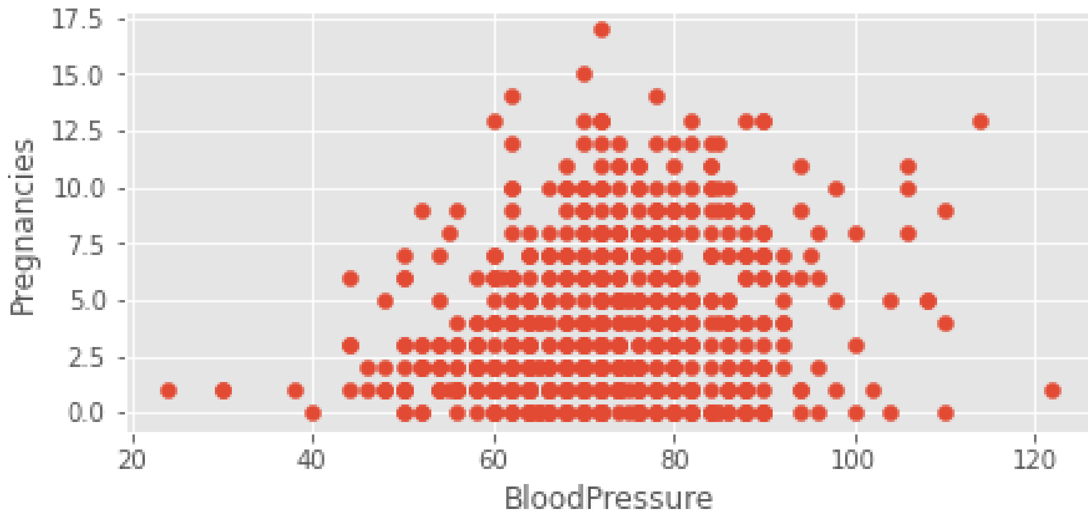
Age vs Glucose



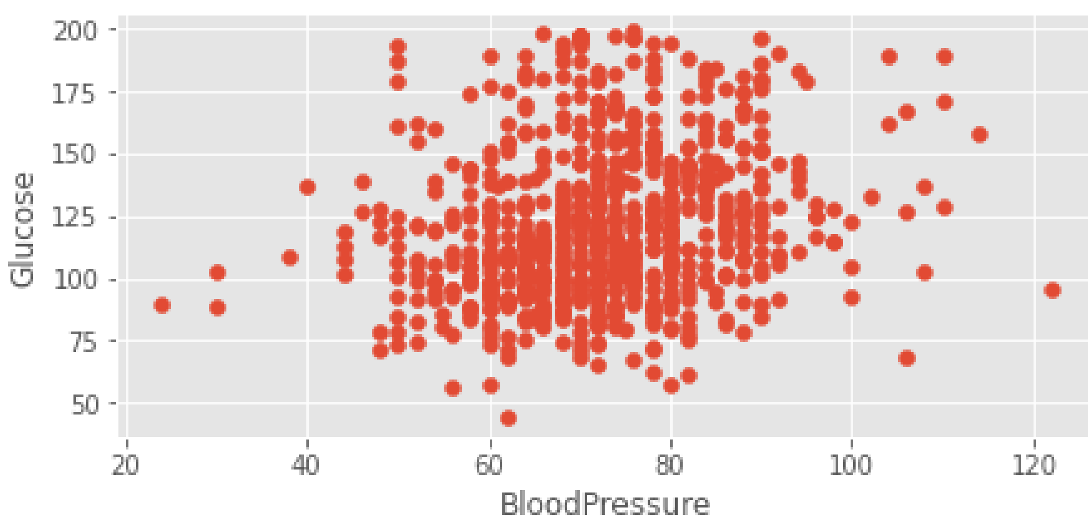
`Outcome vs Glucose`



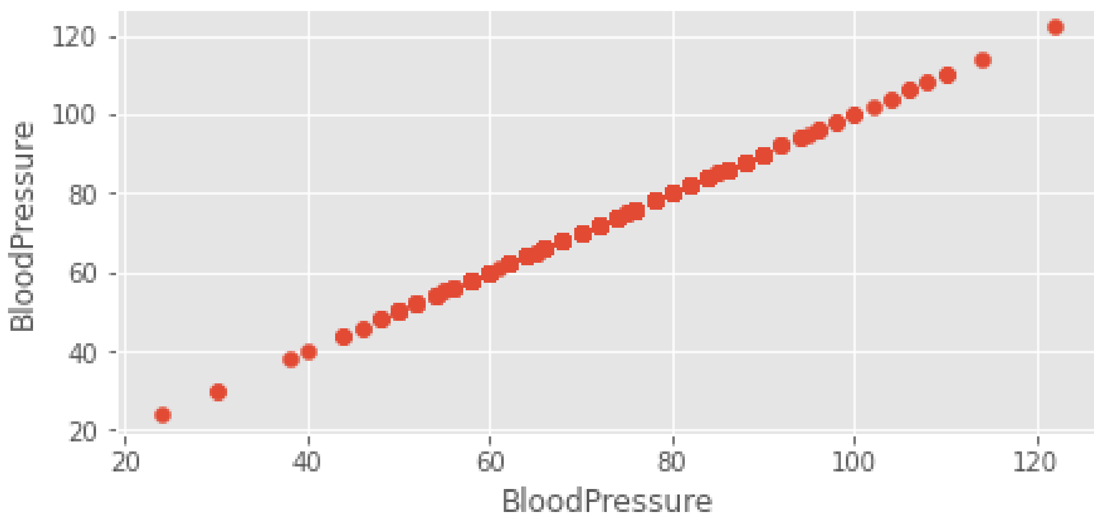
`Pregnancies vs BloodPressure`



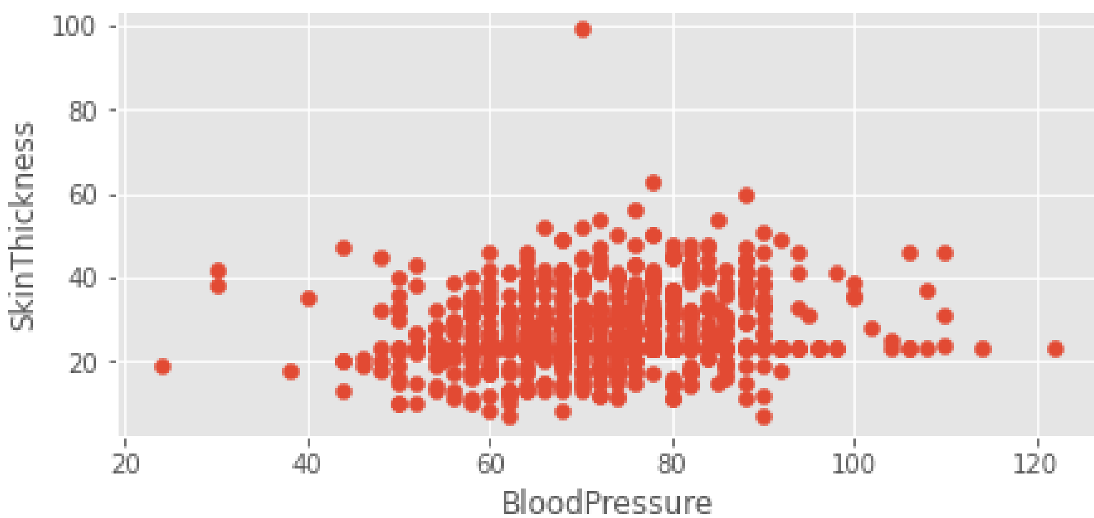
Glucose vs BloodPressure



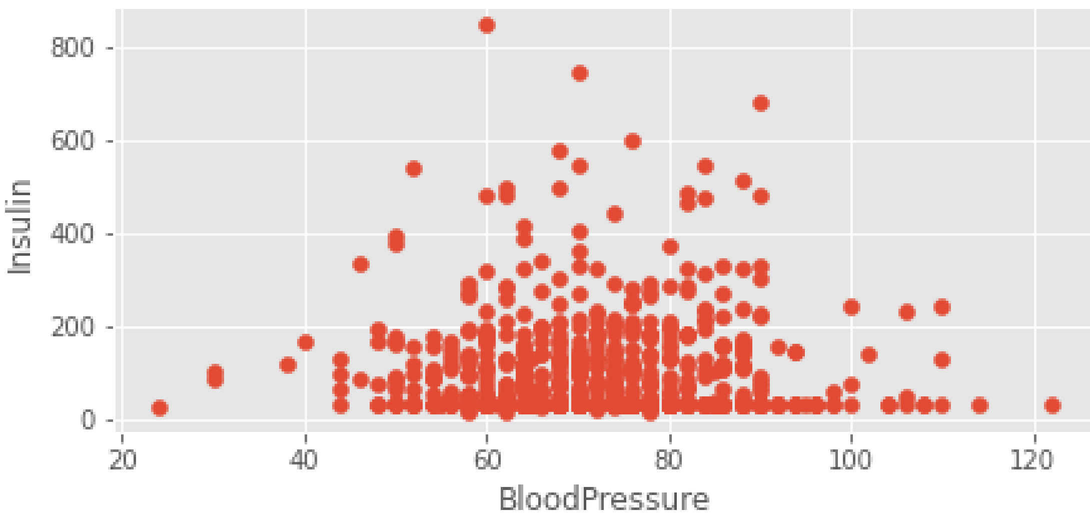
BloodPressure vs BloodPressure



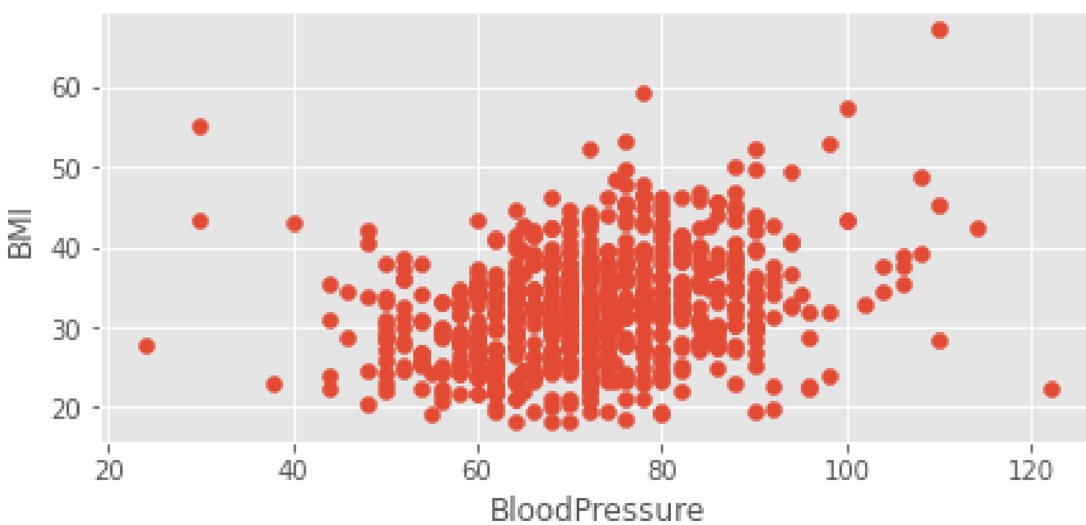
SkinThickness vs BloodPressure



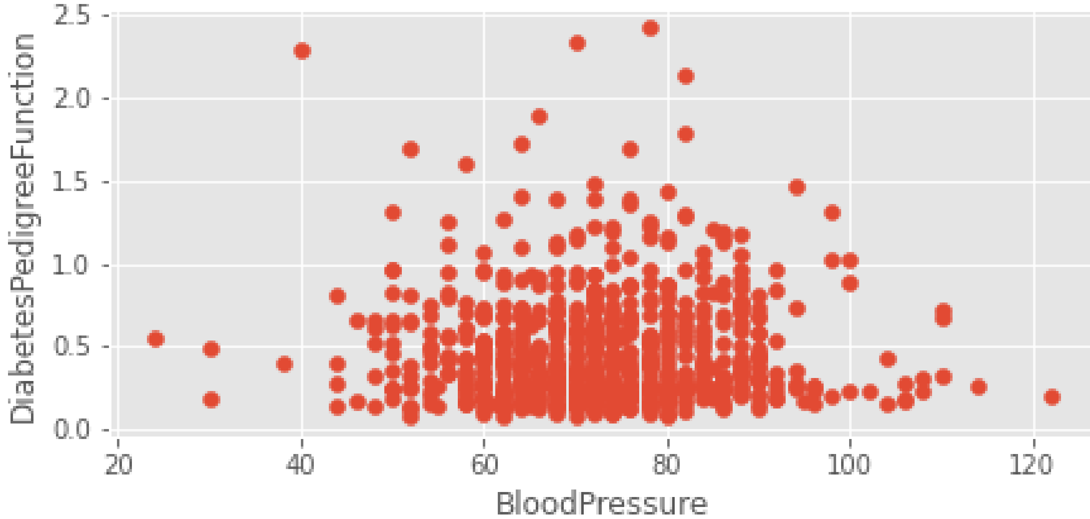
Insulin vs BloodPressure



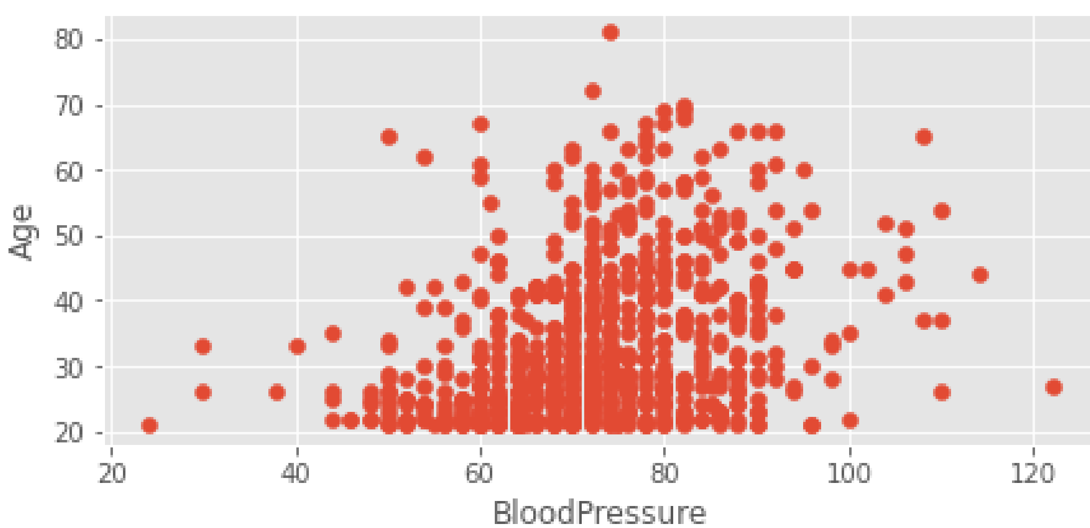

```
BMI vs BloodPressure
```



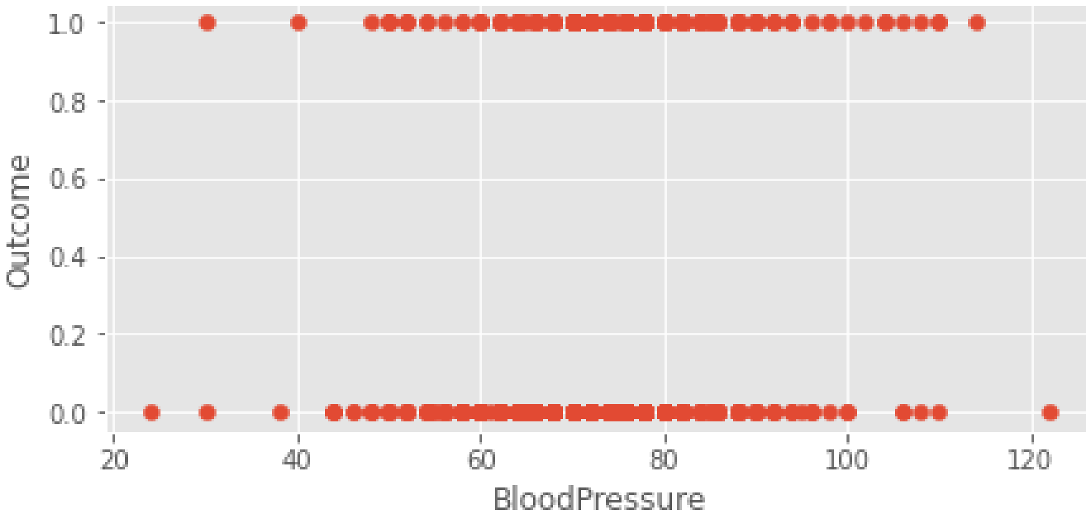

```
DiabetesPedigreeFunction vs BloodPressure
```



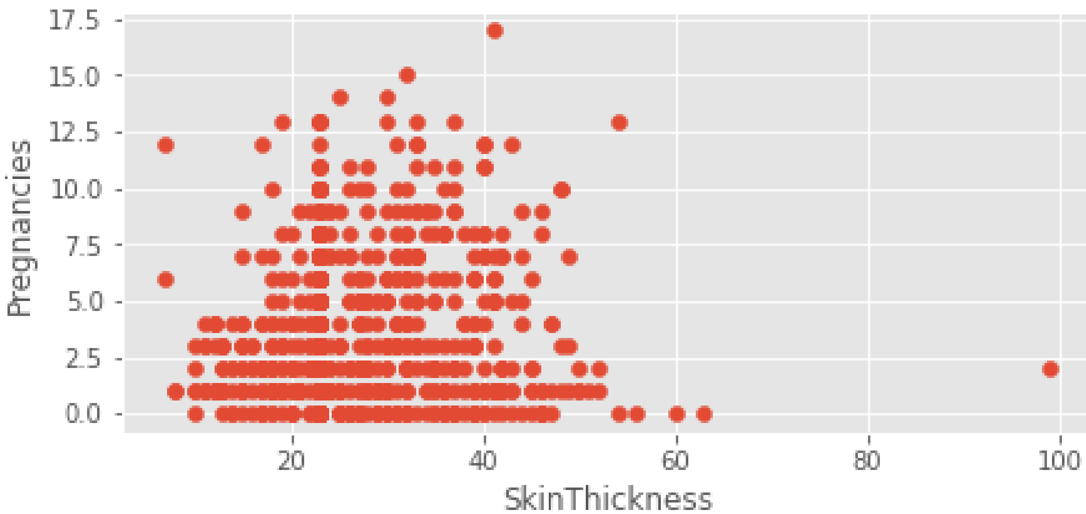

```
Age vs BloodPressure
```



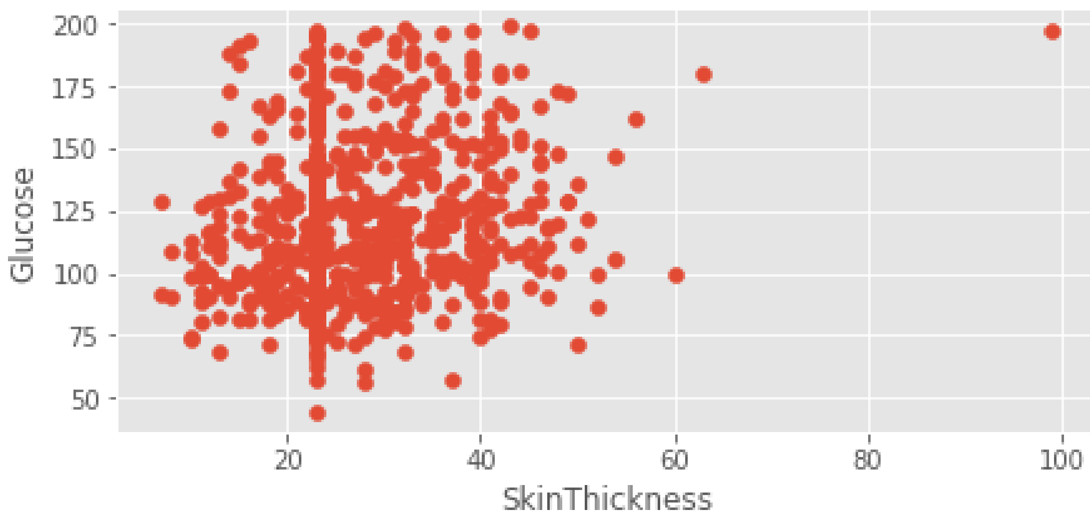

```
Outcome vs BloodPressure
```



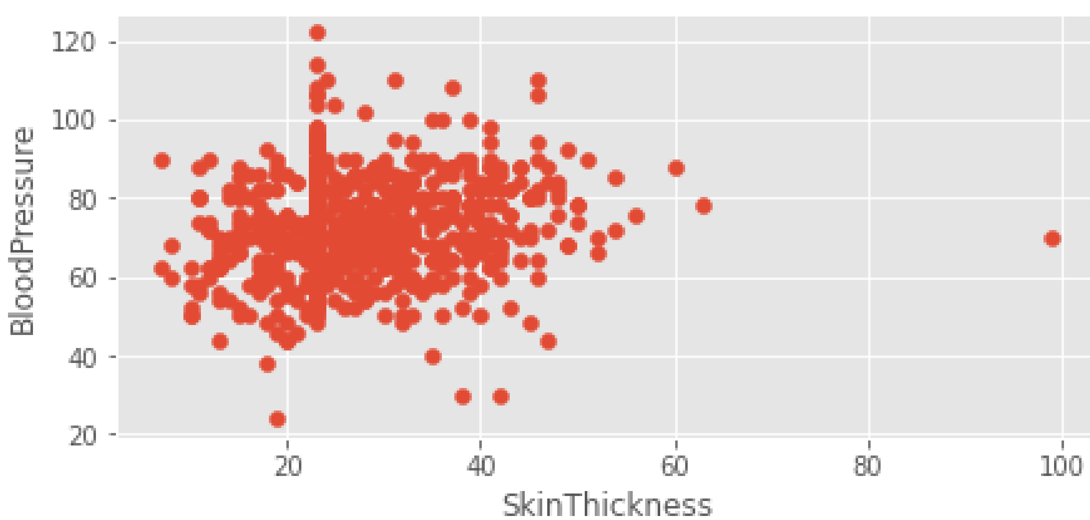
Pregnancies vs SkinThickness



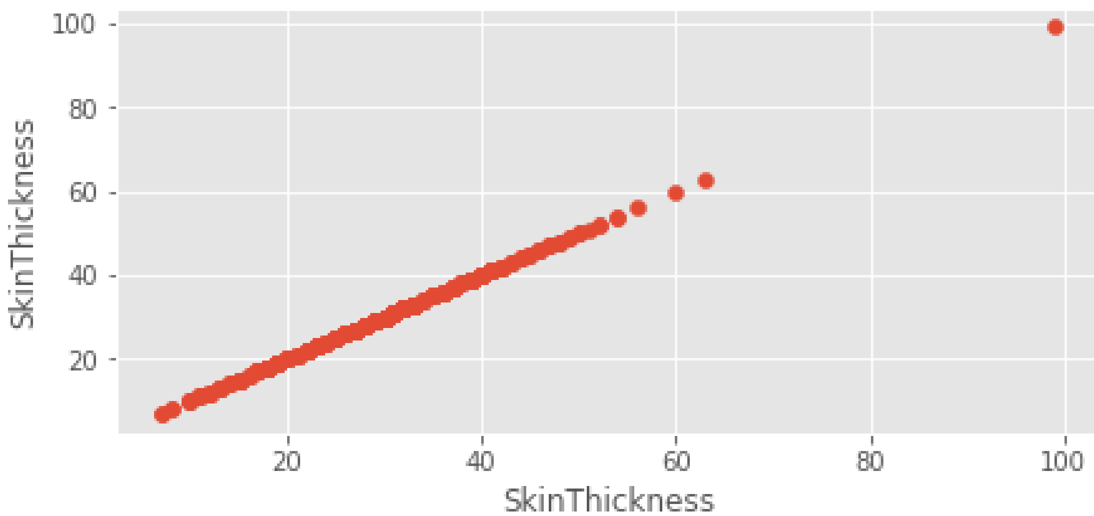
Glucose vs SkinThickness



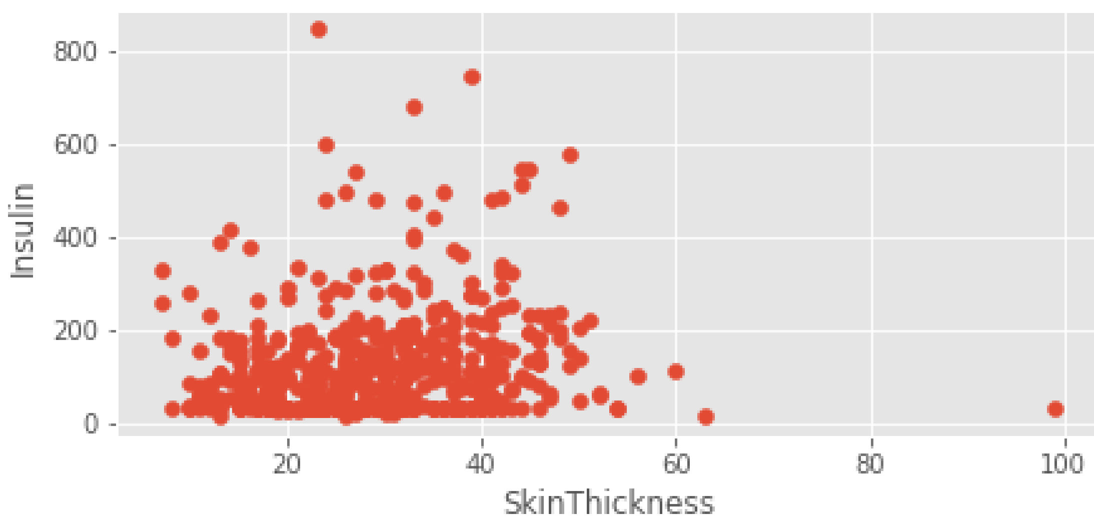
BloodPressure vs SkinThickness



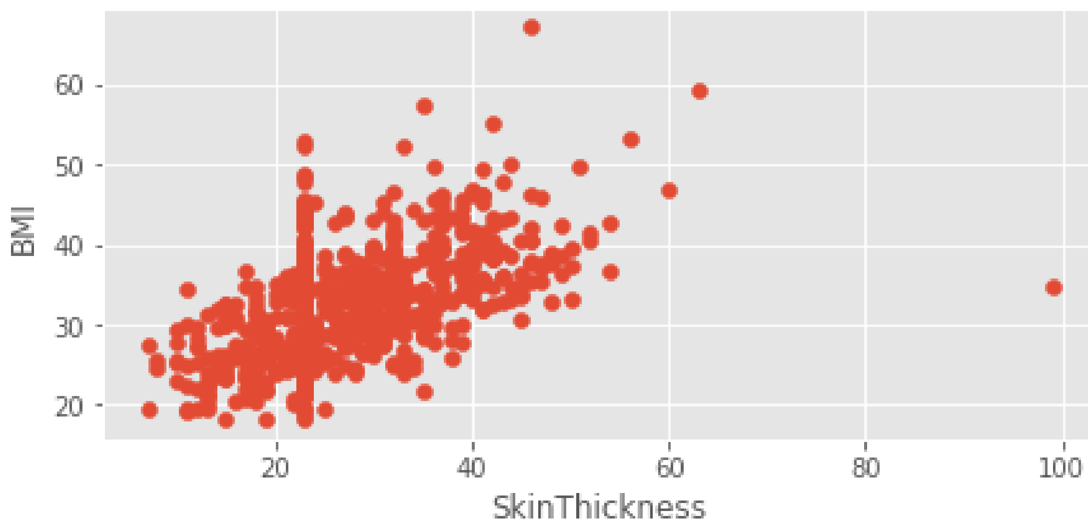
SkinThickness vs SkinThickness



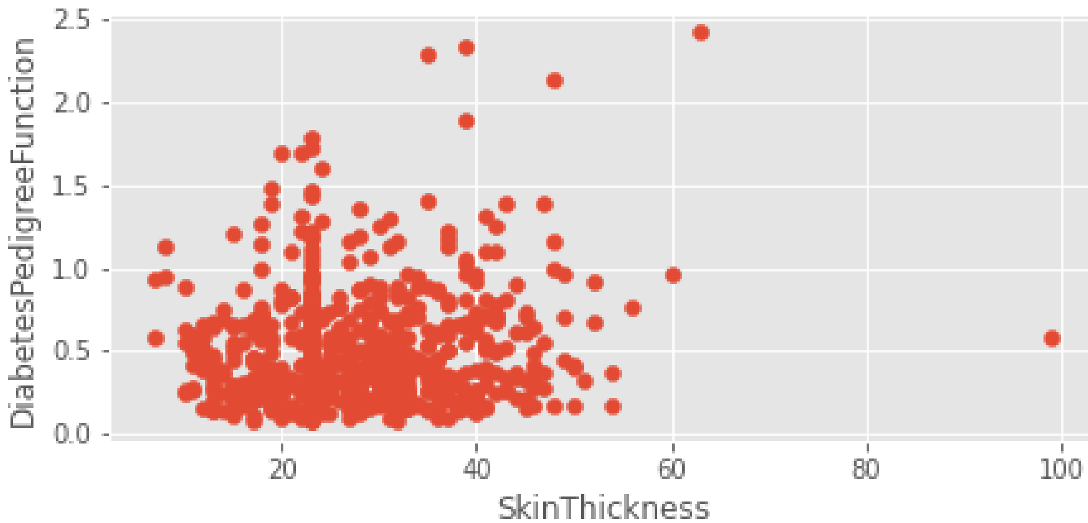
Insulin vs SkinThickness



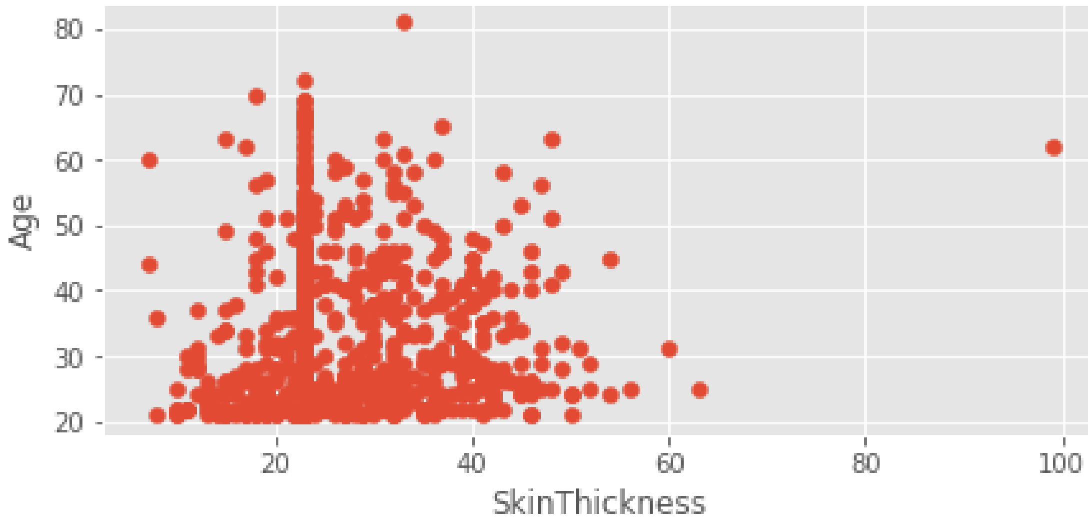
BMI vs SkinThickness



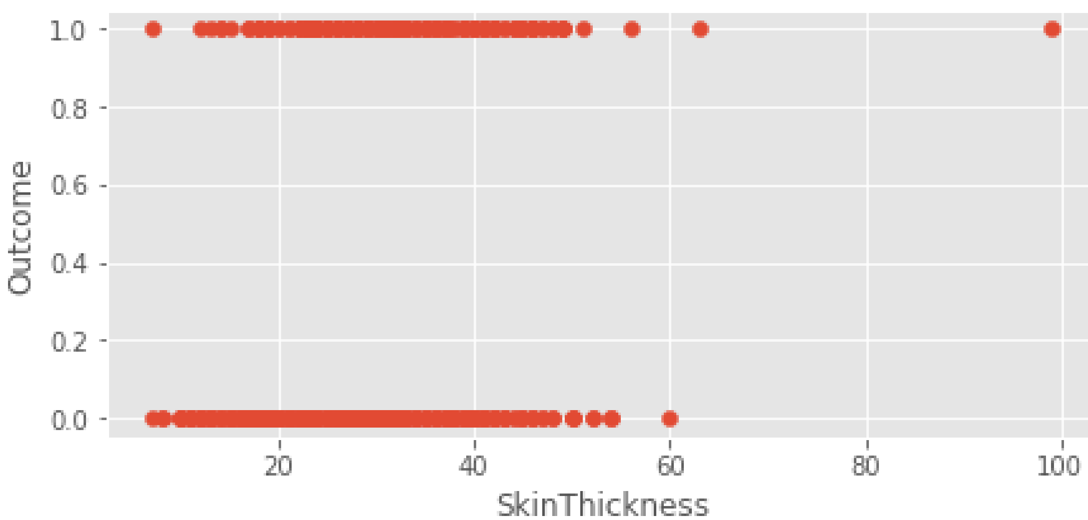
DiabetesPedigreeFunction vs SkinThickness



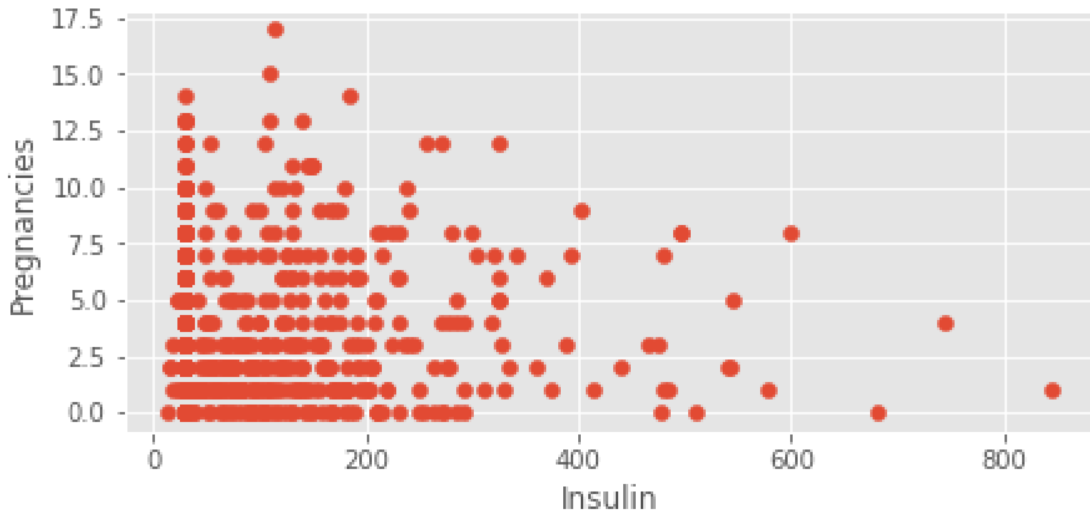
Age vs SkinThickness



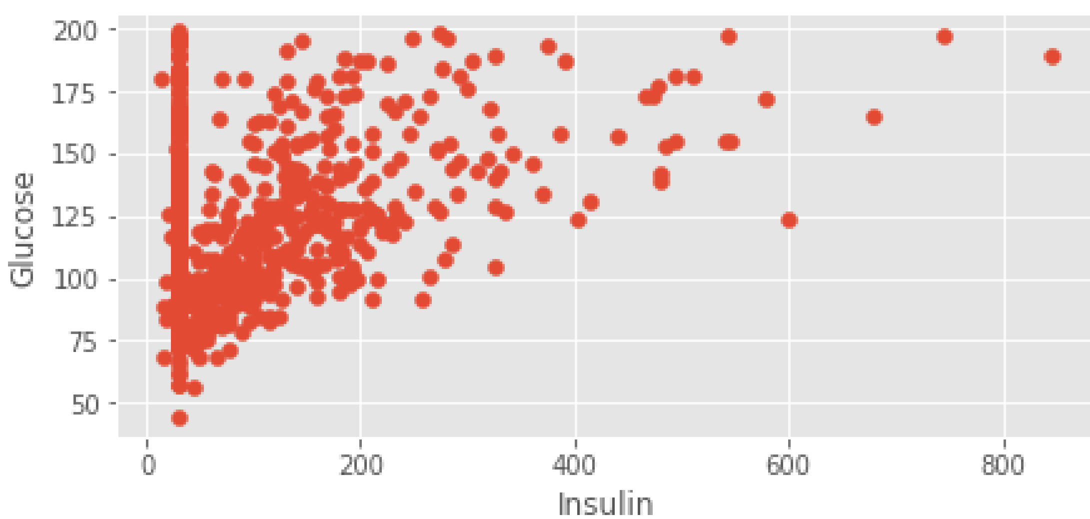
Outcome vs SkinThickness



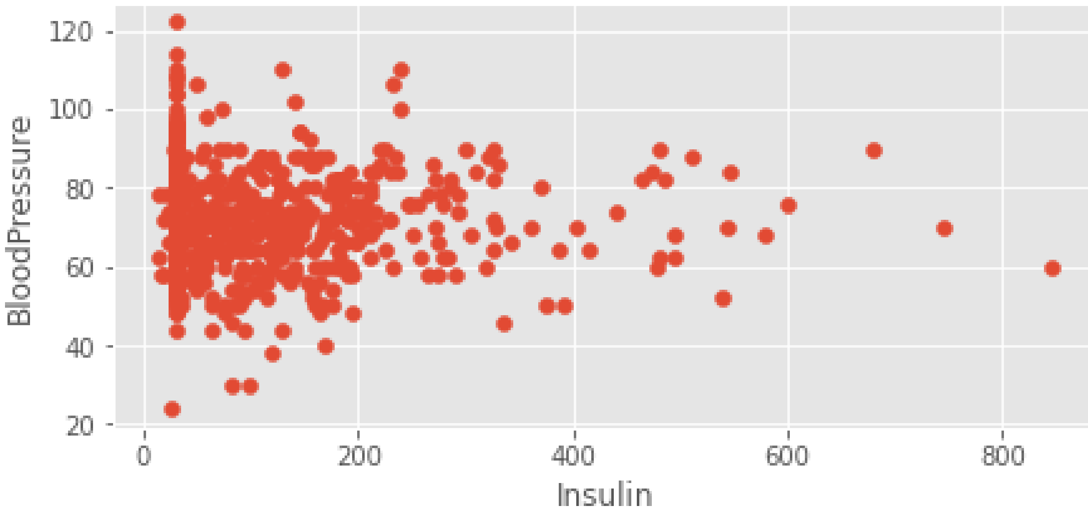
Pregnancies vs Insulin



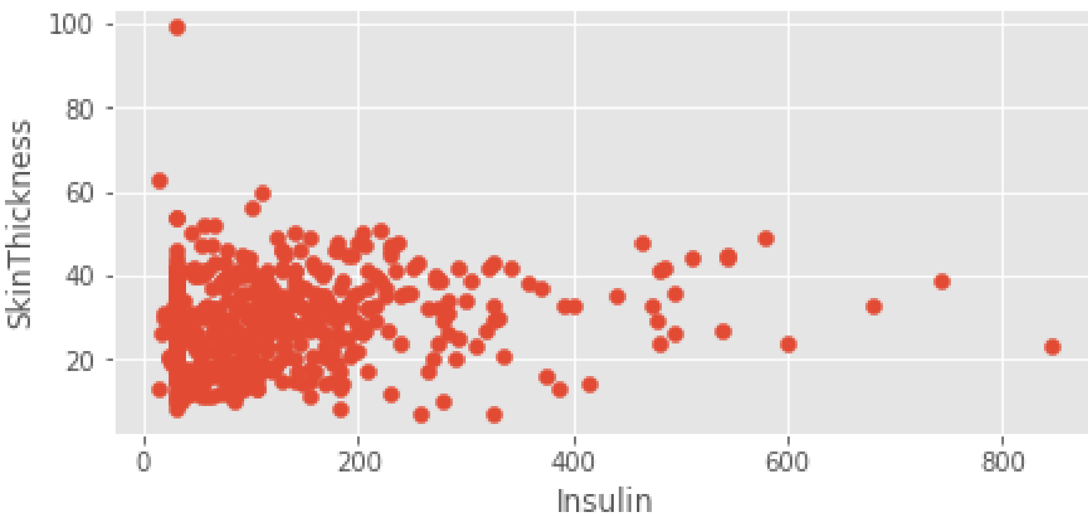
Glucose vs Insulin



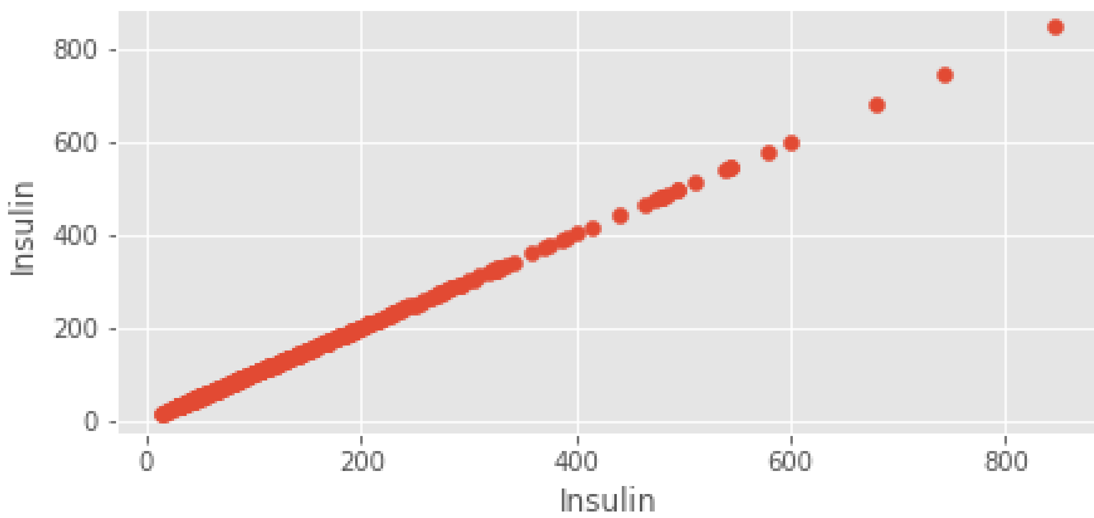
BloodPressure vs Insulin



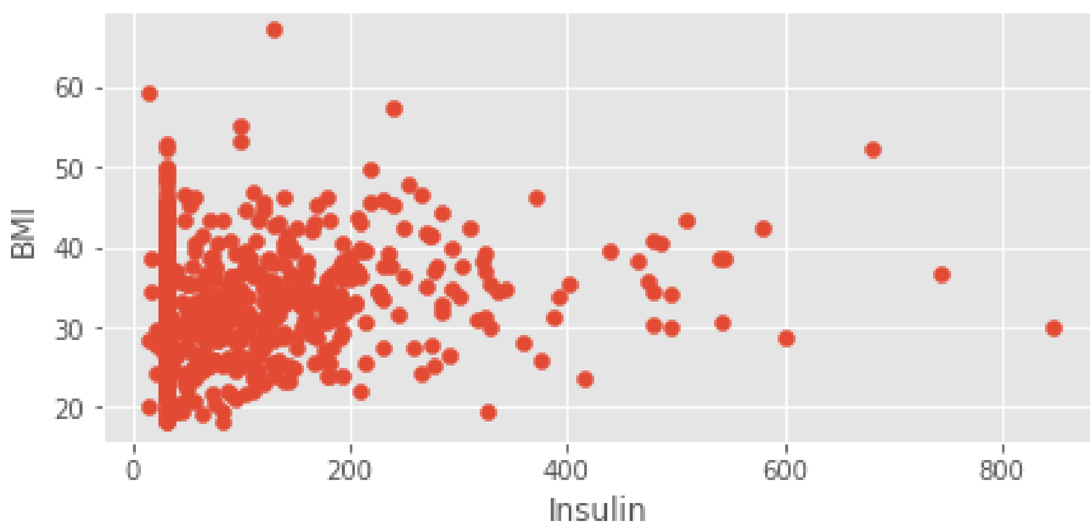
SkinThickness vs Insulin



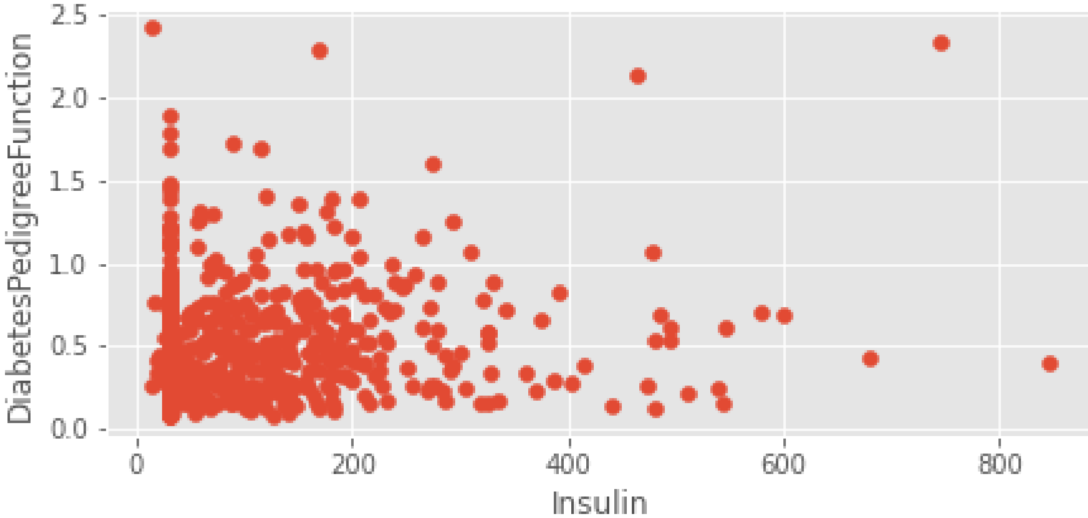
Insulin vs Insulin



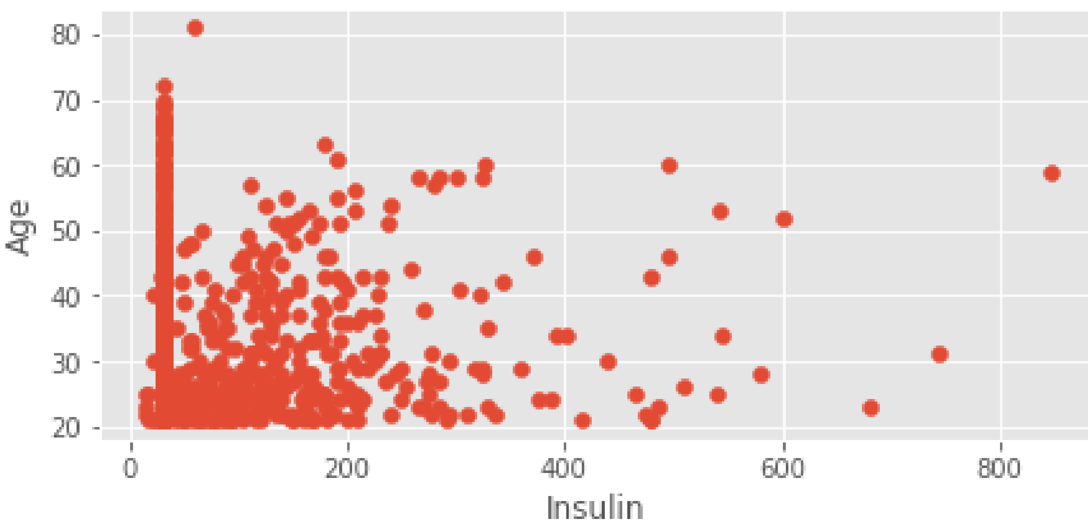
BMI vs Insulin



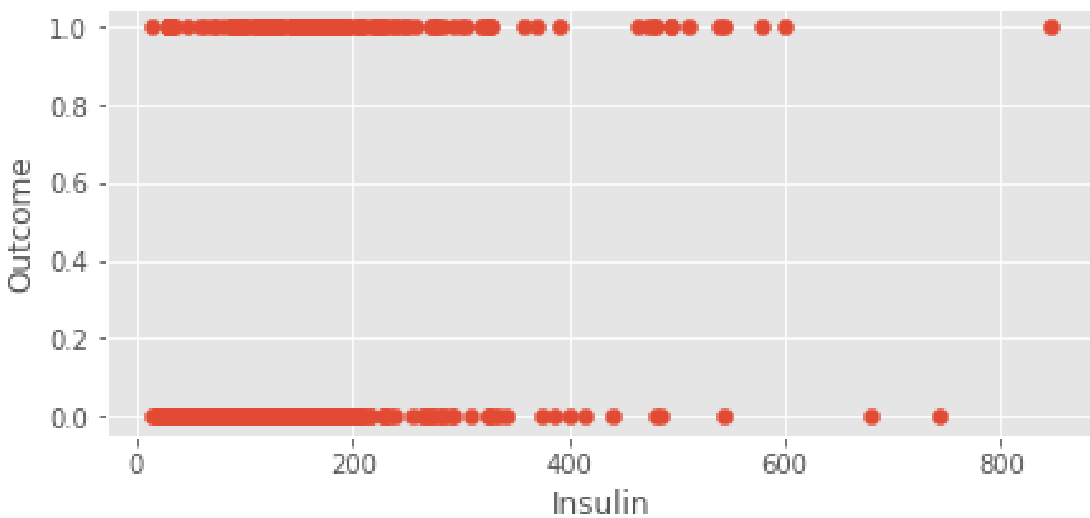
DiabetesPedigreeFunction vs Insulin



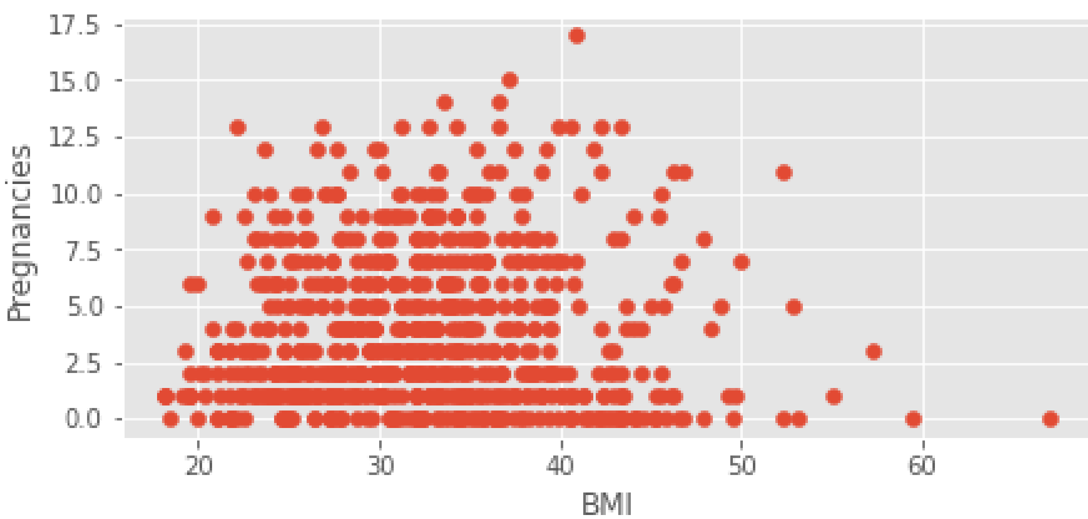
Age vs Insulin



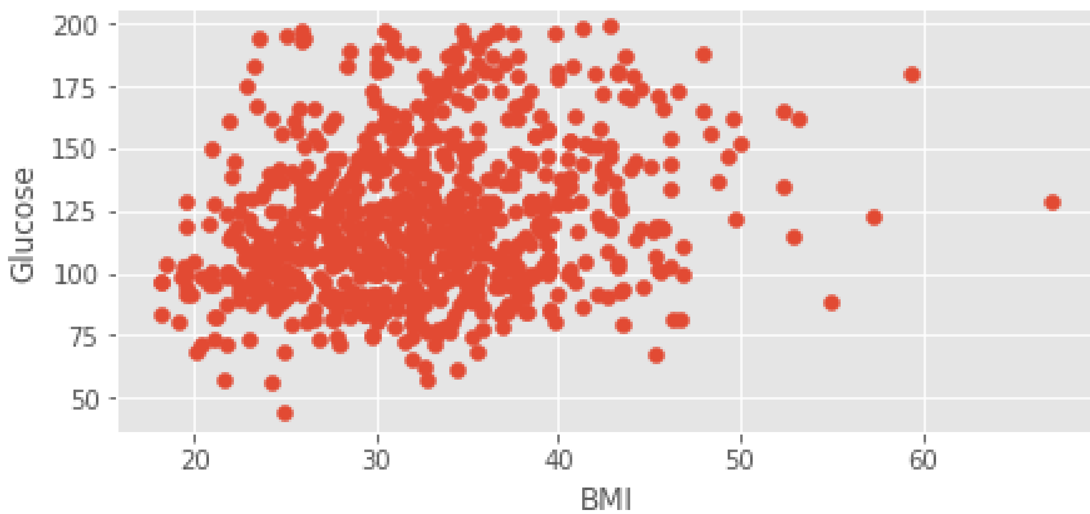
Outcome vs Insulin



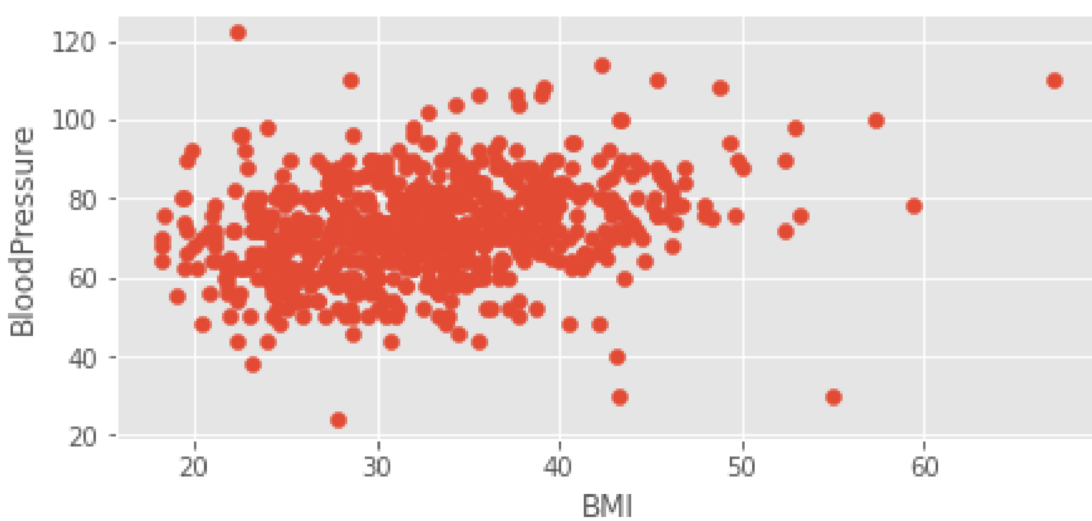
Pregnancies vs BMI



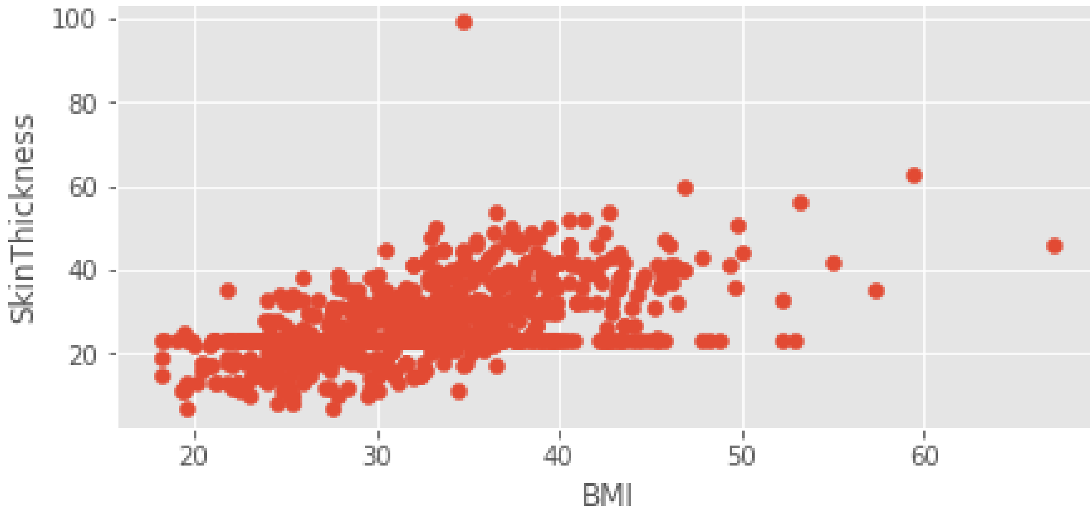
Glucose vs BMI



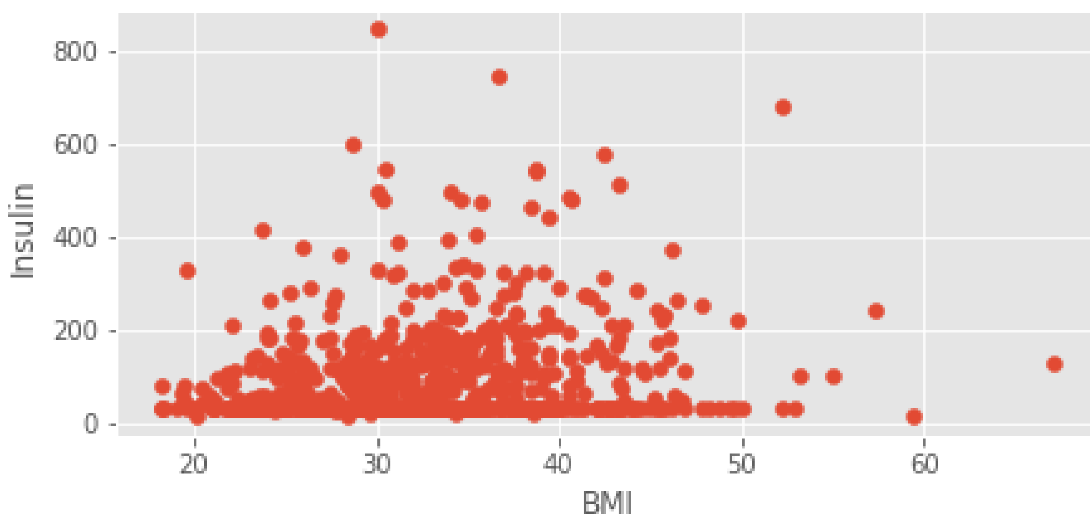
BloodPressure vs BMI



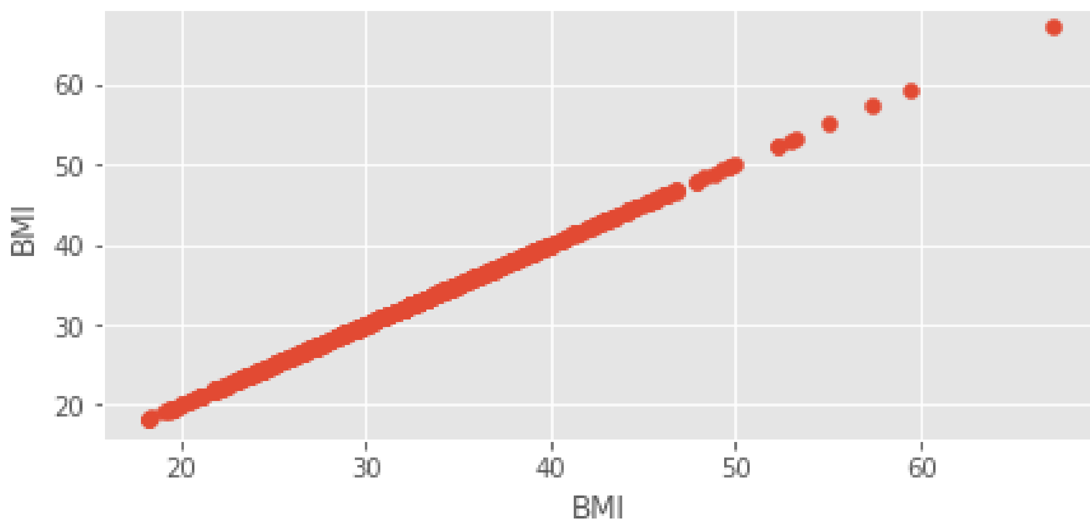
SkinThickness vs BMI



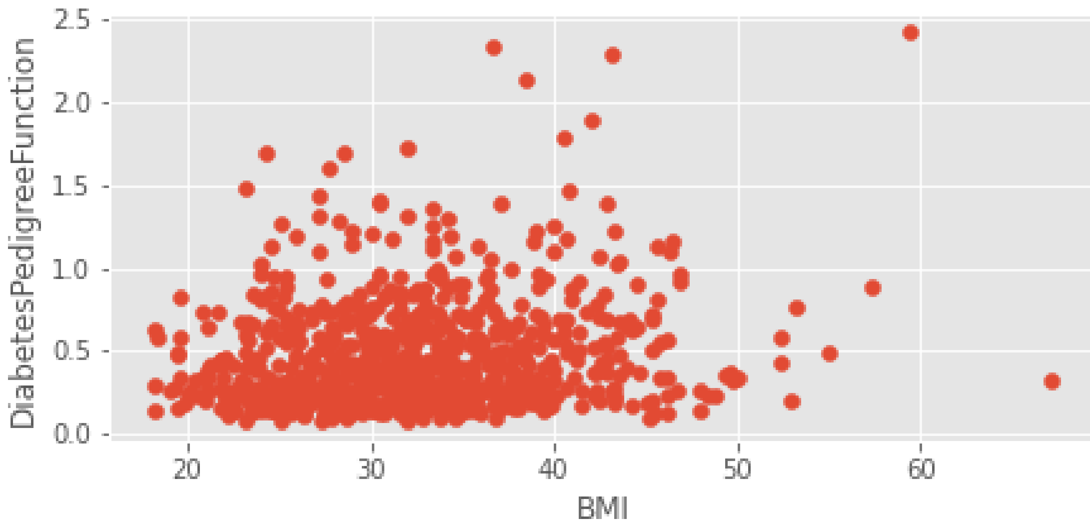
Insulin vs BMI



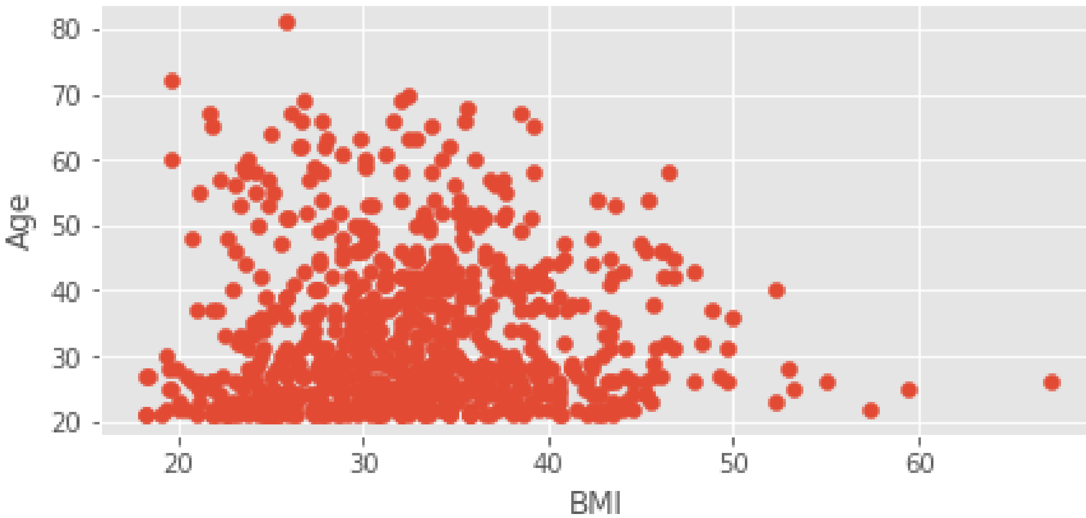
BMI vs BMI



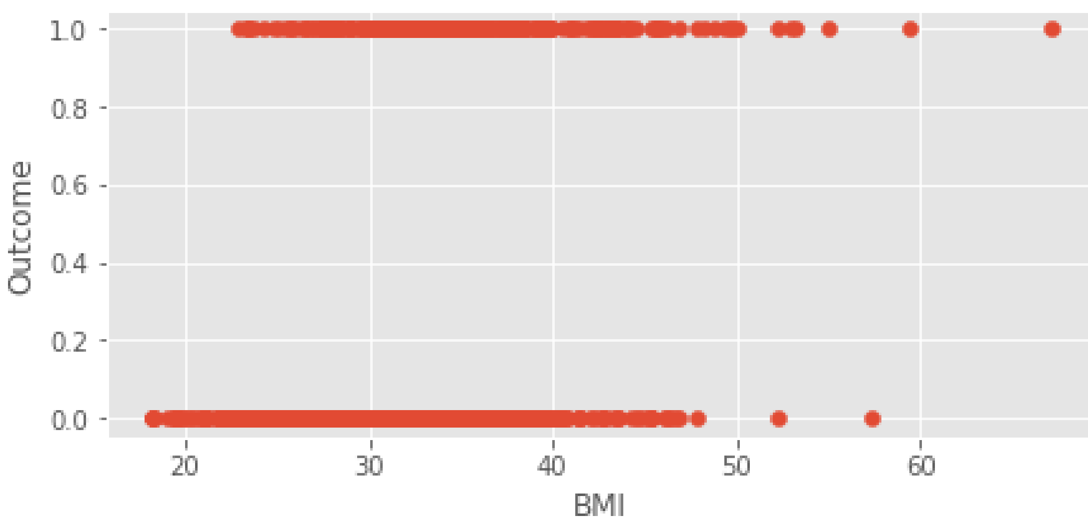
DiabetesPedigreeFunction vs BMI



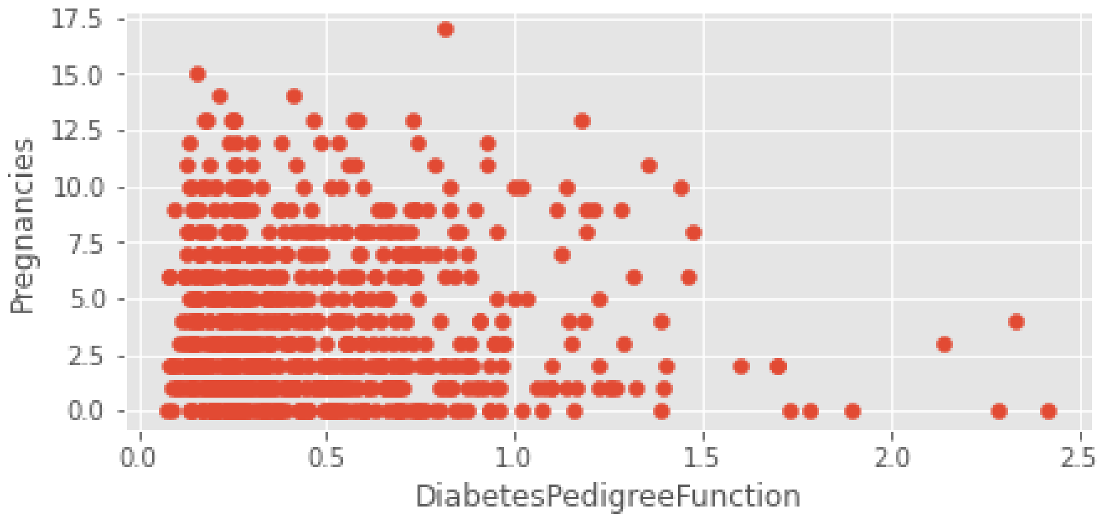
Age vs BMI



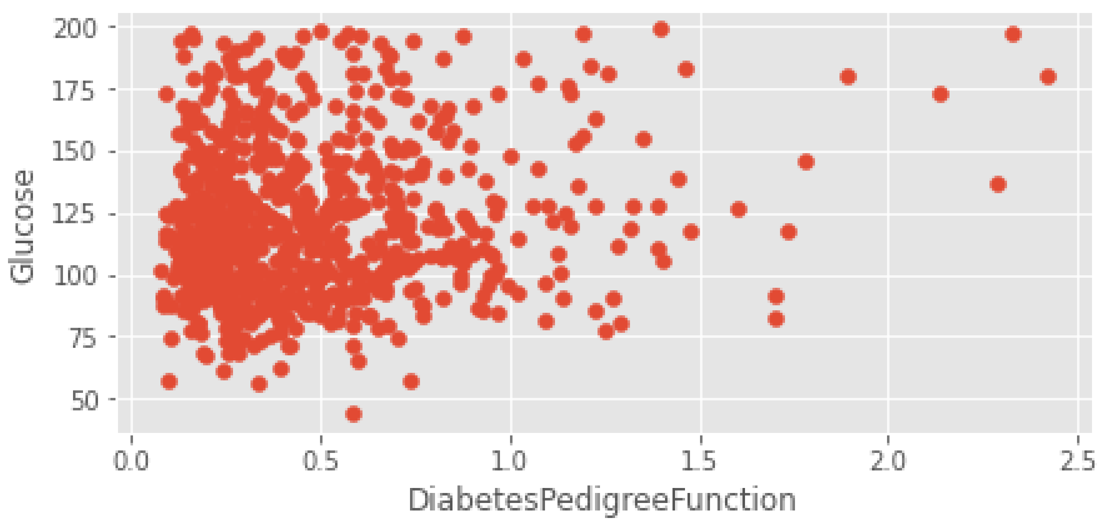
Outcome vs BMI



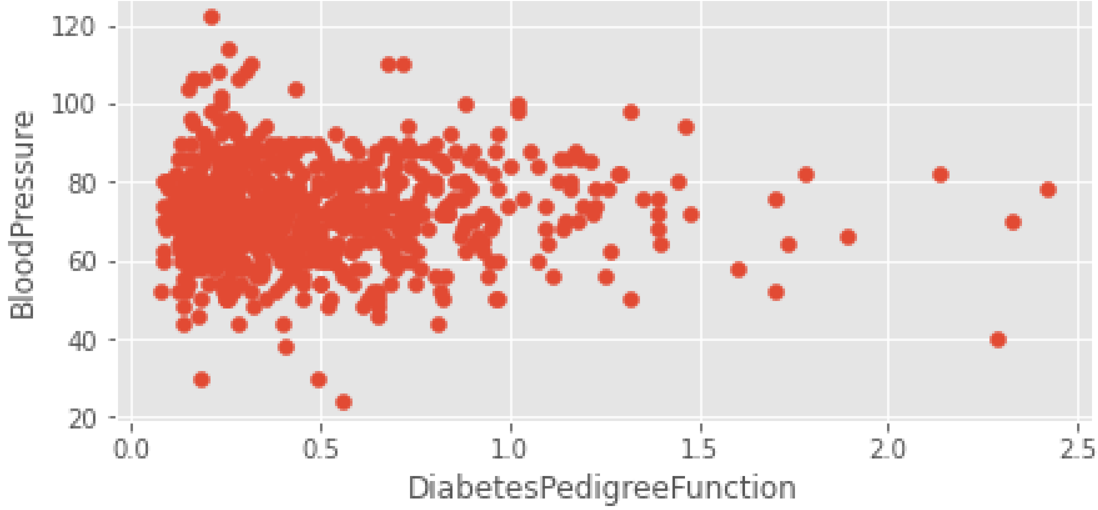
Pregnancies vs DiabetesPedigreeFunction



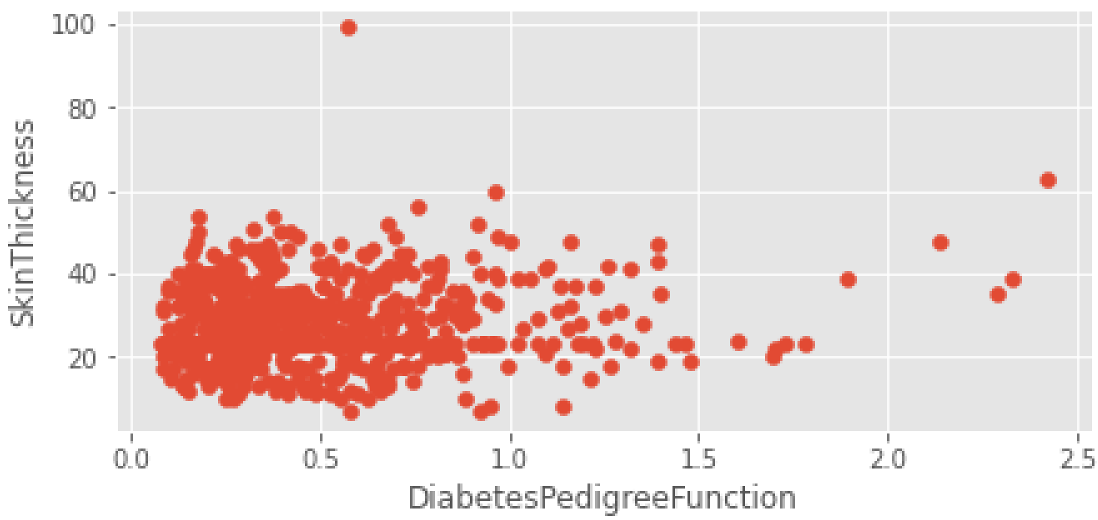
Glucose vs DiabetesPedigreeFunction



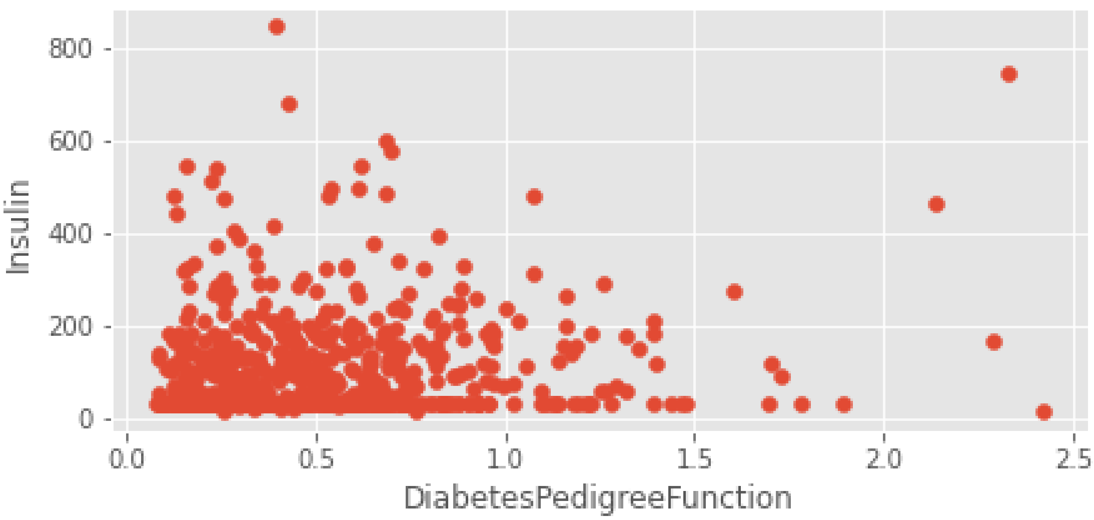
BloodPressure vs DiabetesPedigreeFunction



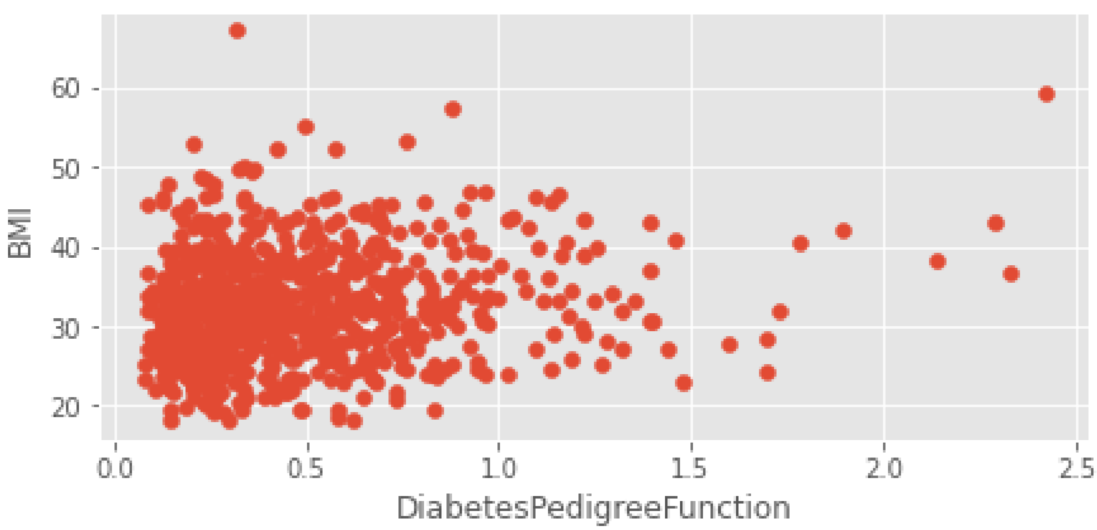
SkinThickness vs DiabetesPedigreeFunction



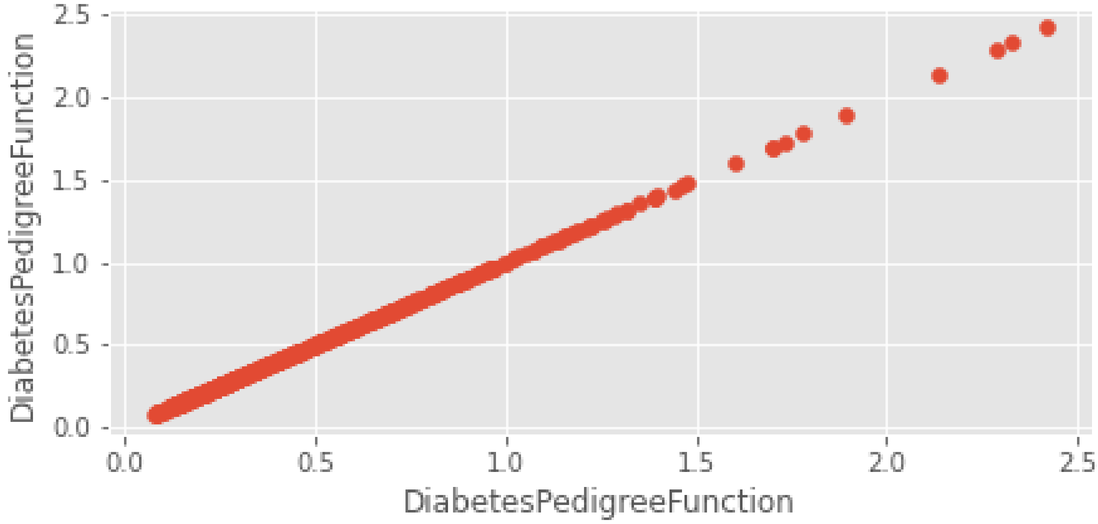
Insulin vs DiabetesPedigreeFunction



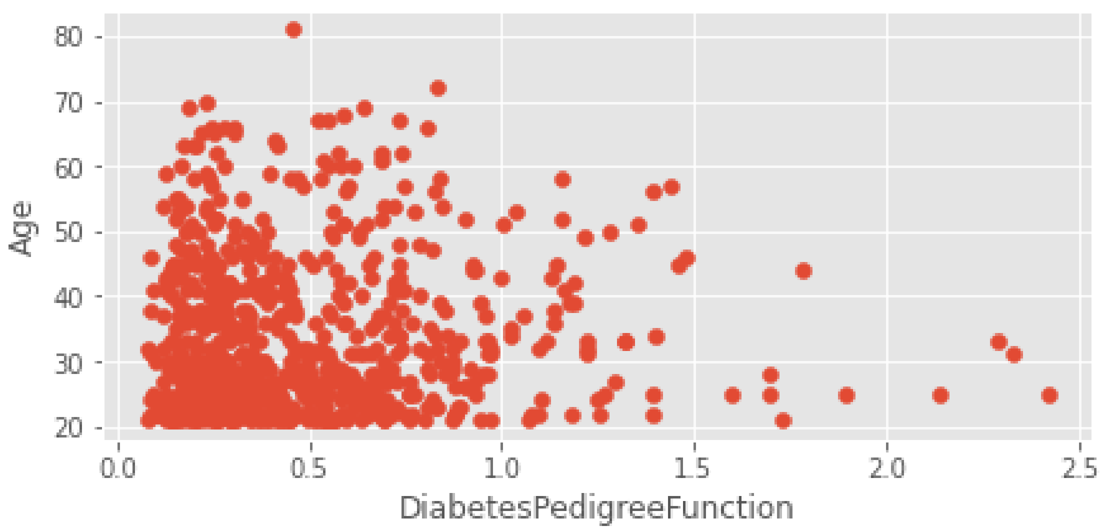
BMI vs DiabetesPedigreeFunction



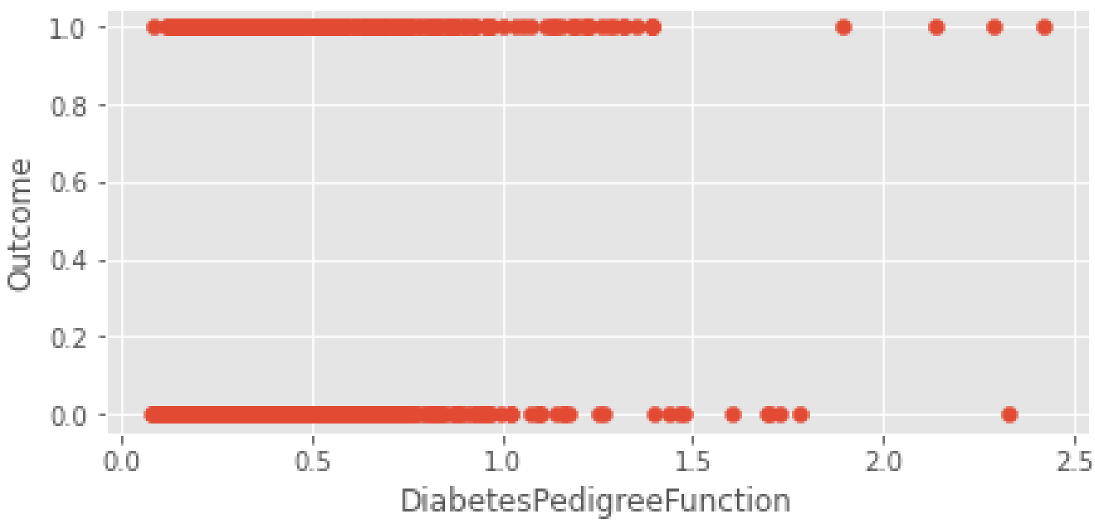
DiabetesPedigreeFunction vs DiabetesPedigreeFunction



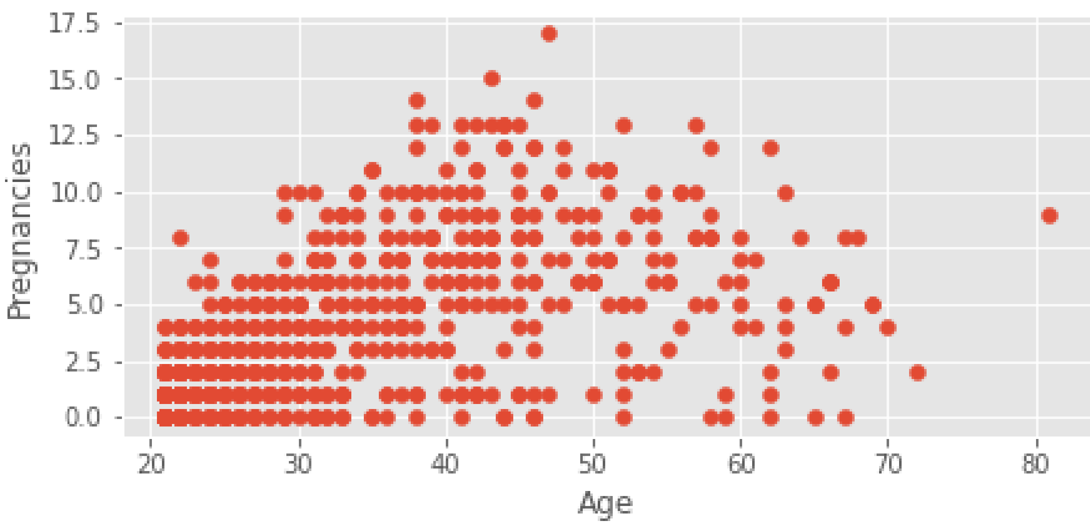
Age vs DiabetesPedigreeFunction



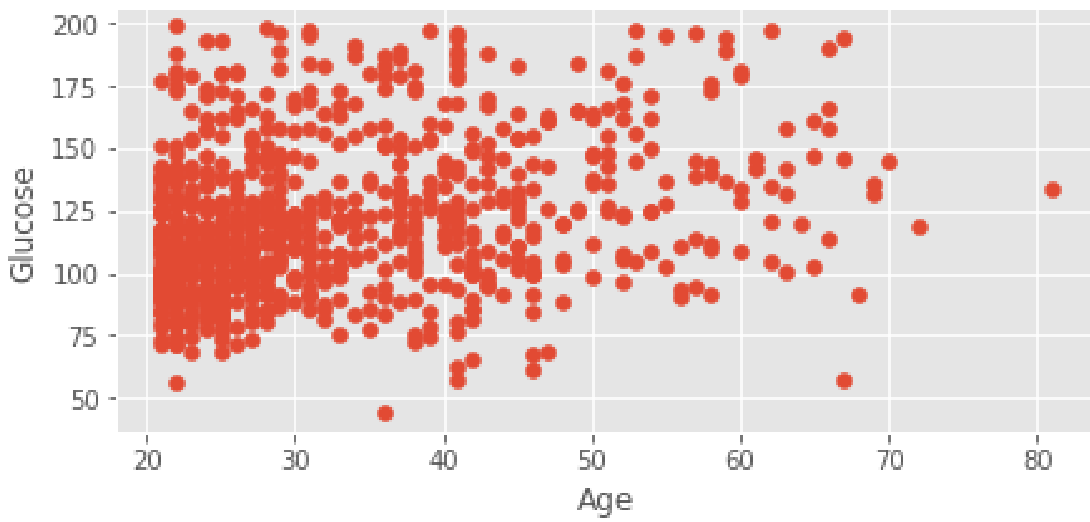
Outcome vs DiabetesPedigreeFunction



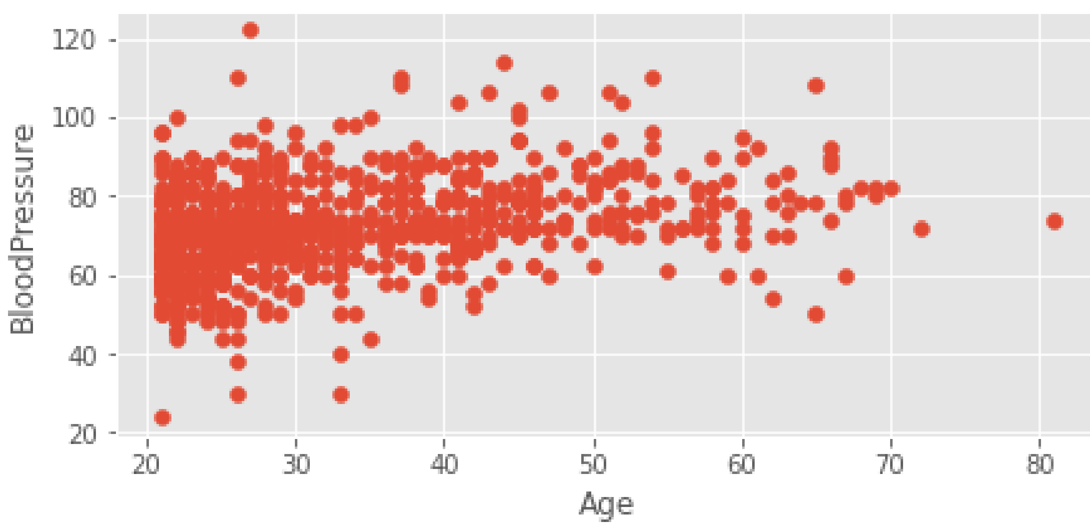
Pregnancies vs Age



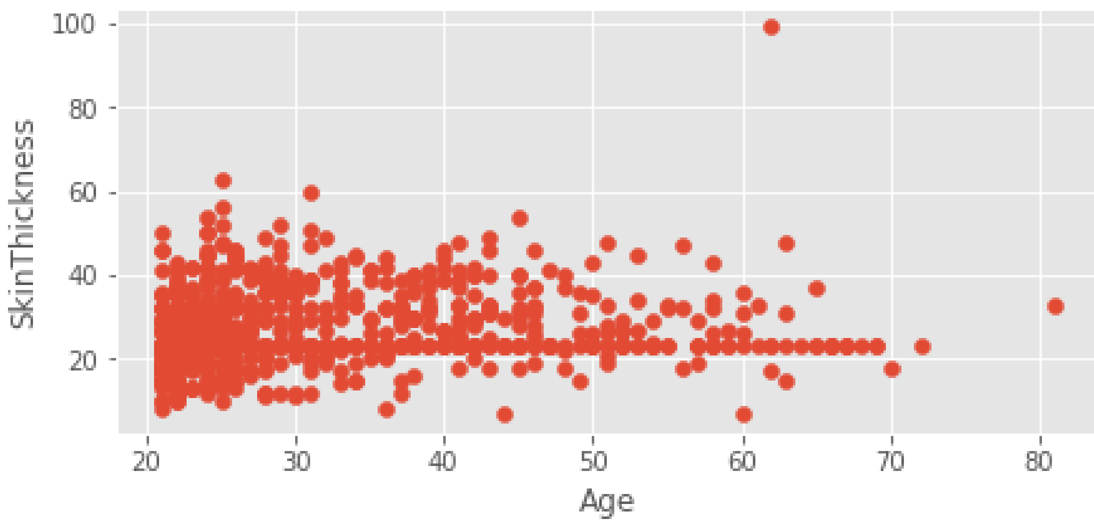
Glucose vs Age



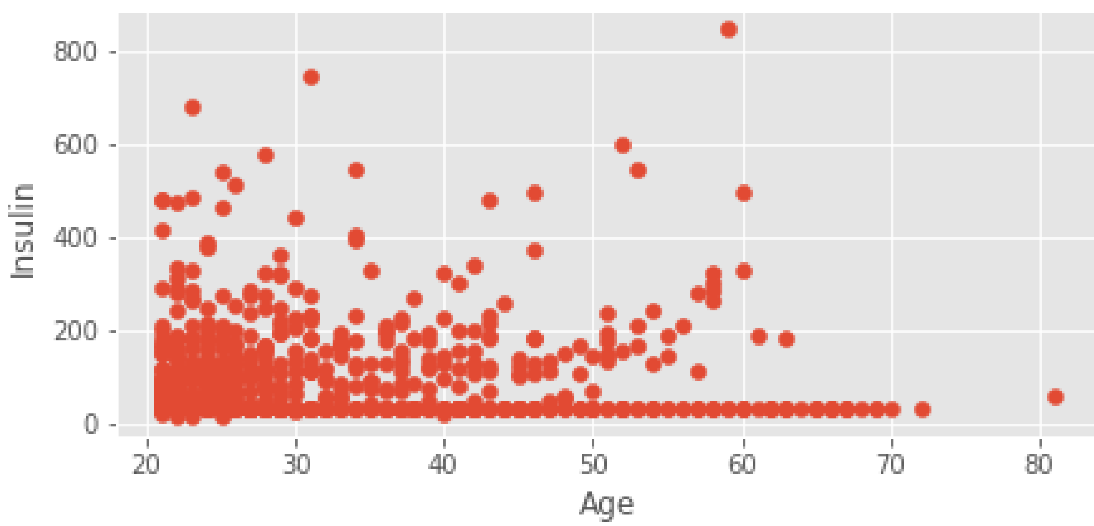
BloodPressure vs Age



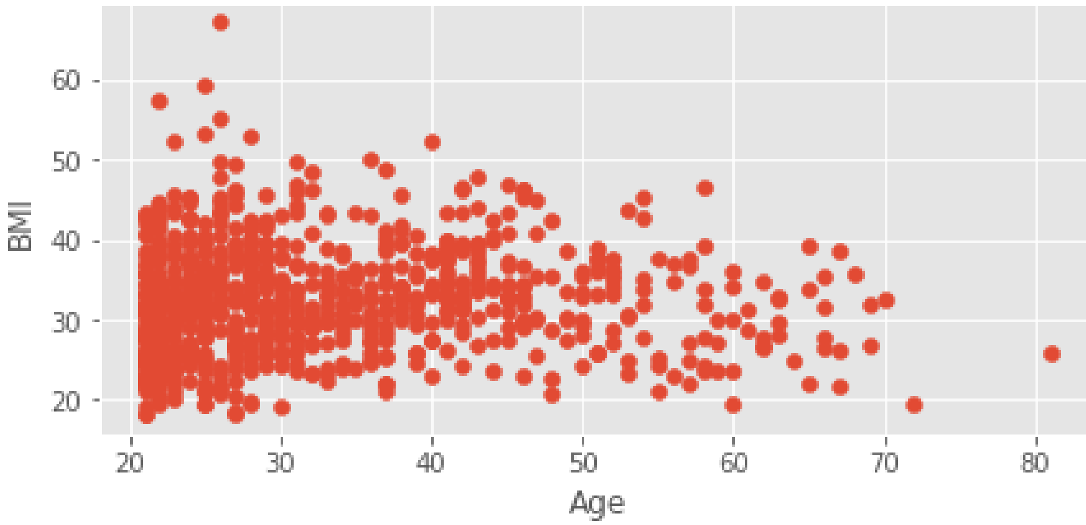
SkinThickness vs Age



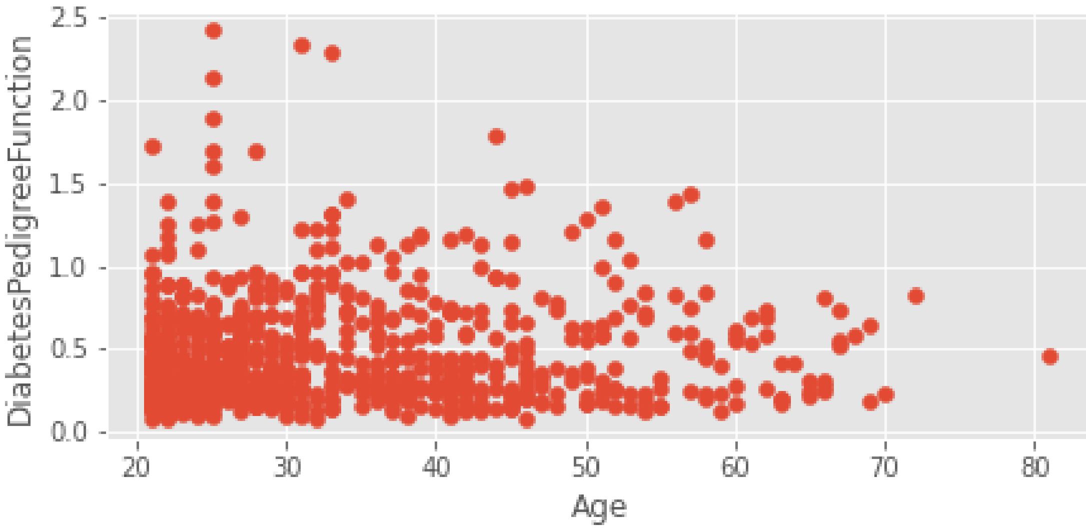
Insulin vs Age



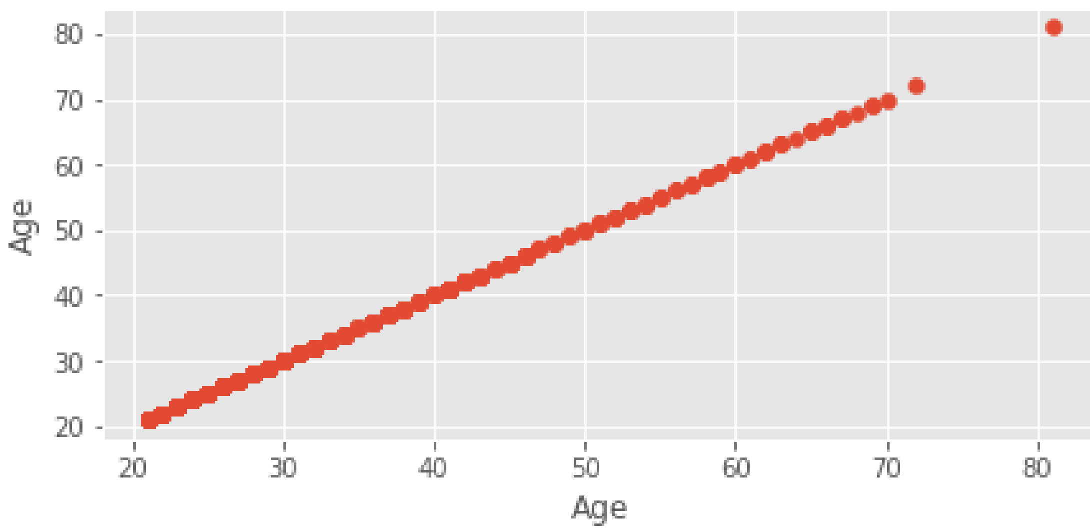
BMI vs Age



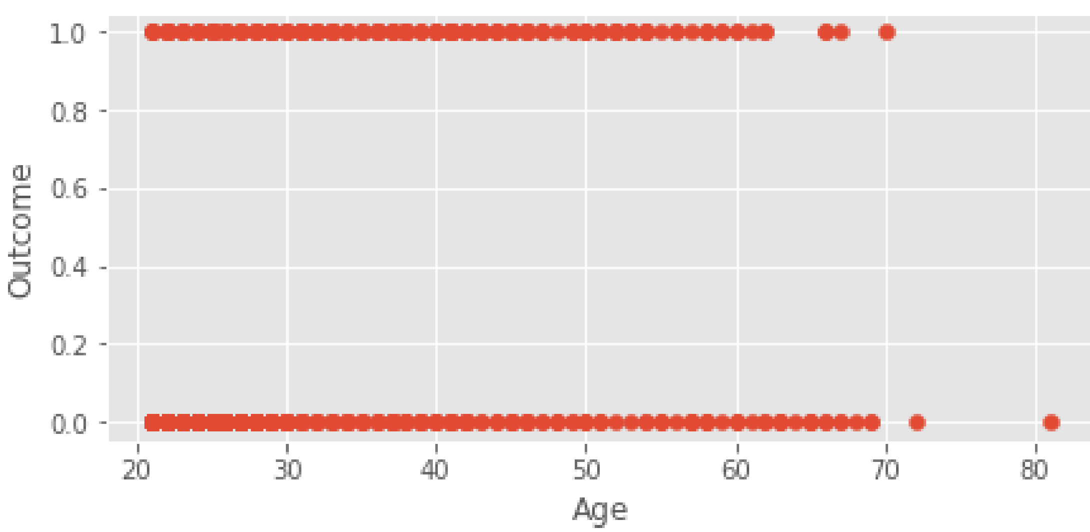
DiabetesPedigreeFunction vs Age



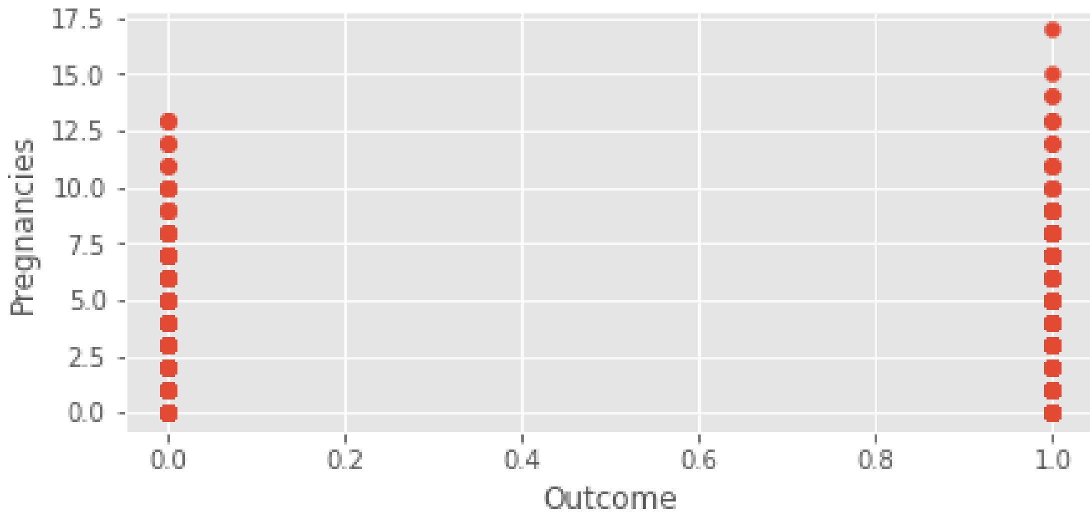
Age vs Age



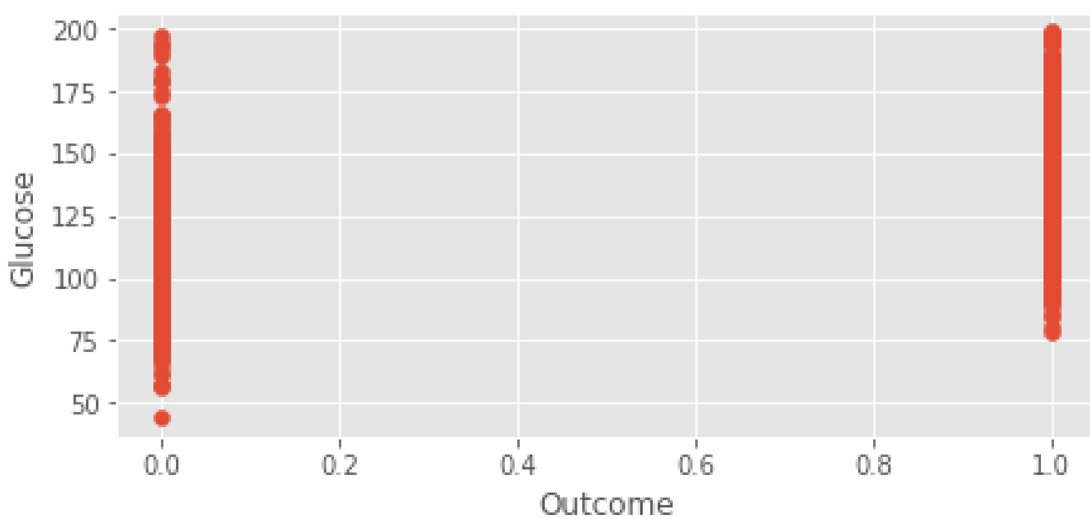
Outcome vs Age



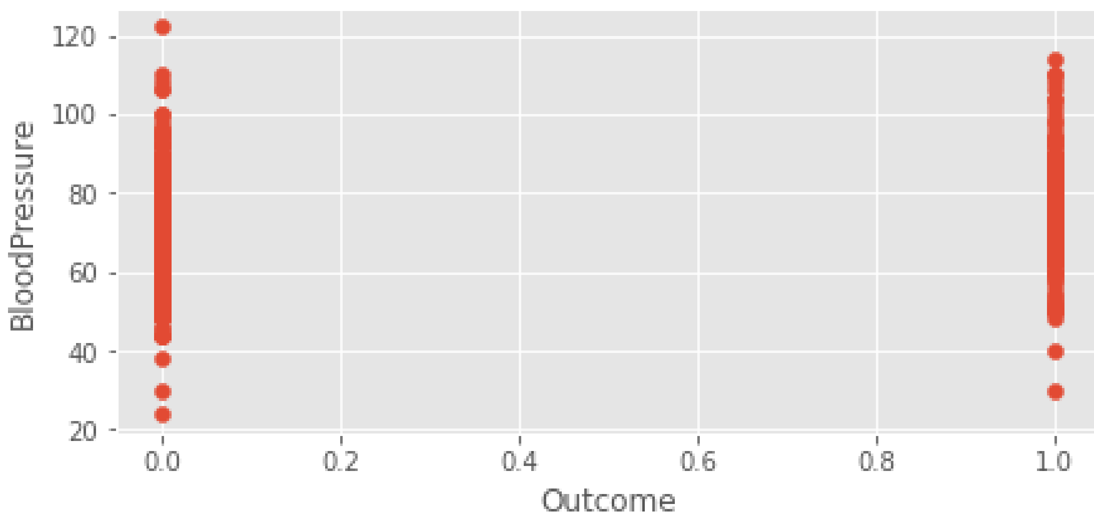
Pregnancies vs Outcome



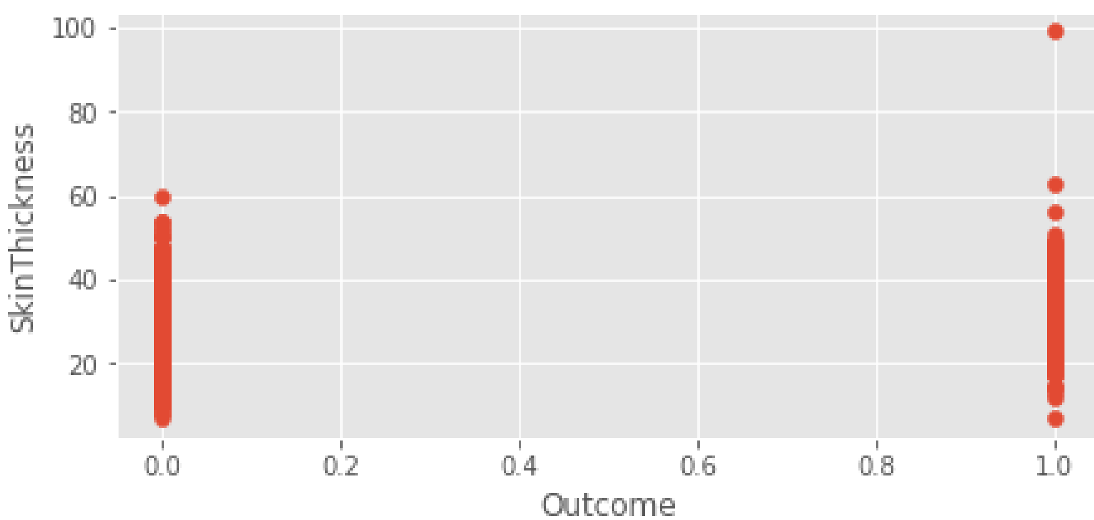
Glucose vs Outcome



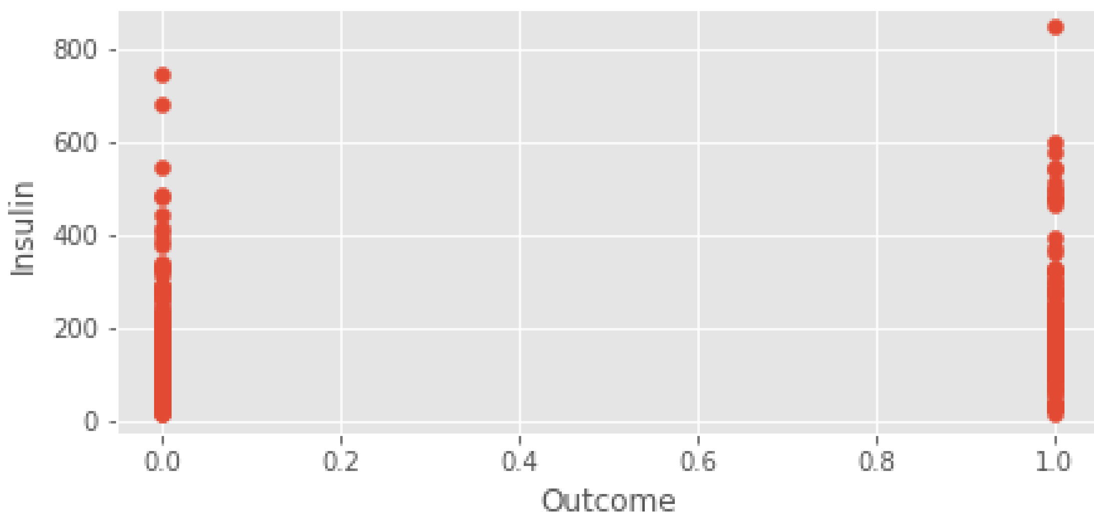
BloodPressure vs Outcome



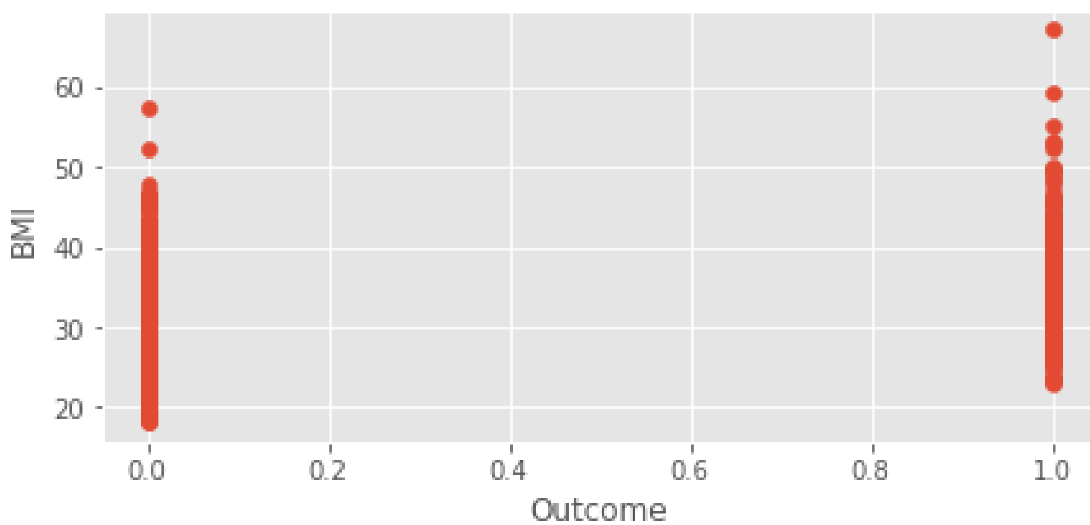
SkinThickness vs Outcome



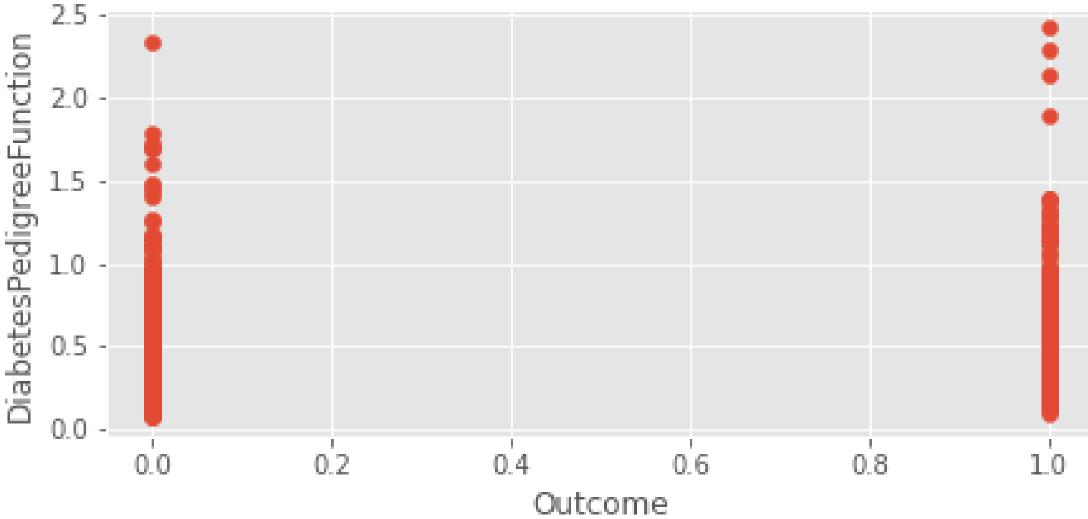
Insulin vs Outcome



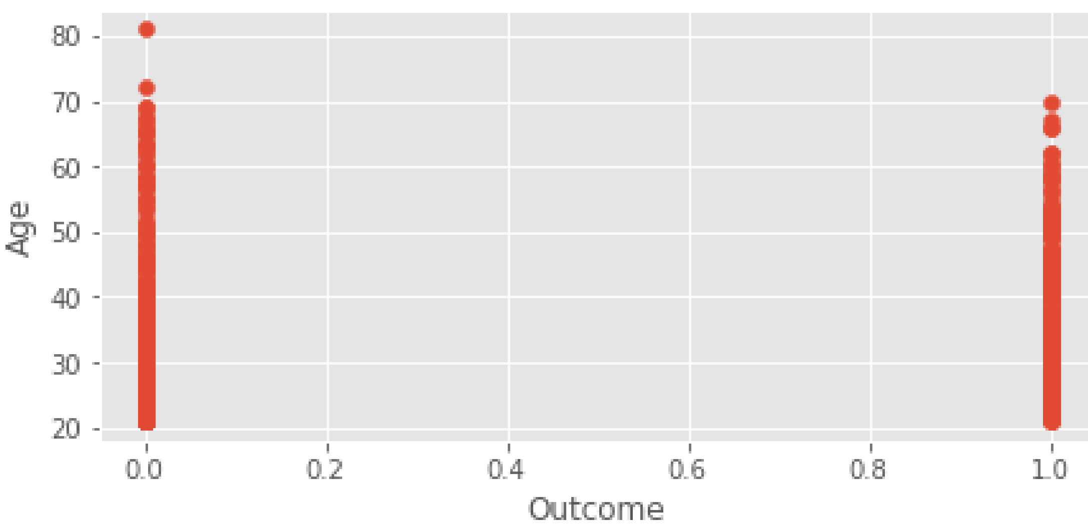
BMI vs Outcome



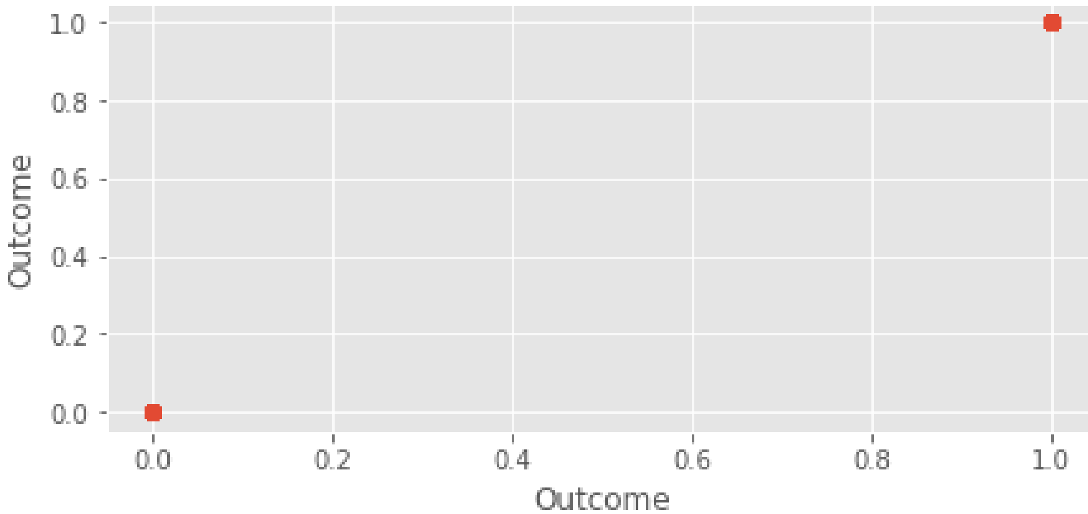
DiabetesPedigreeFunction vs Outcome



Age vs Outcome



Outcome vs Outcome



[27]:
```
# The Above scatter plots show us that most of the variables have one to many␣
↪relations,
#or are independent of each other or have very small relation between them.
```

[28]:
```
#correlation analysis. Visually explore it using a heat map.
```

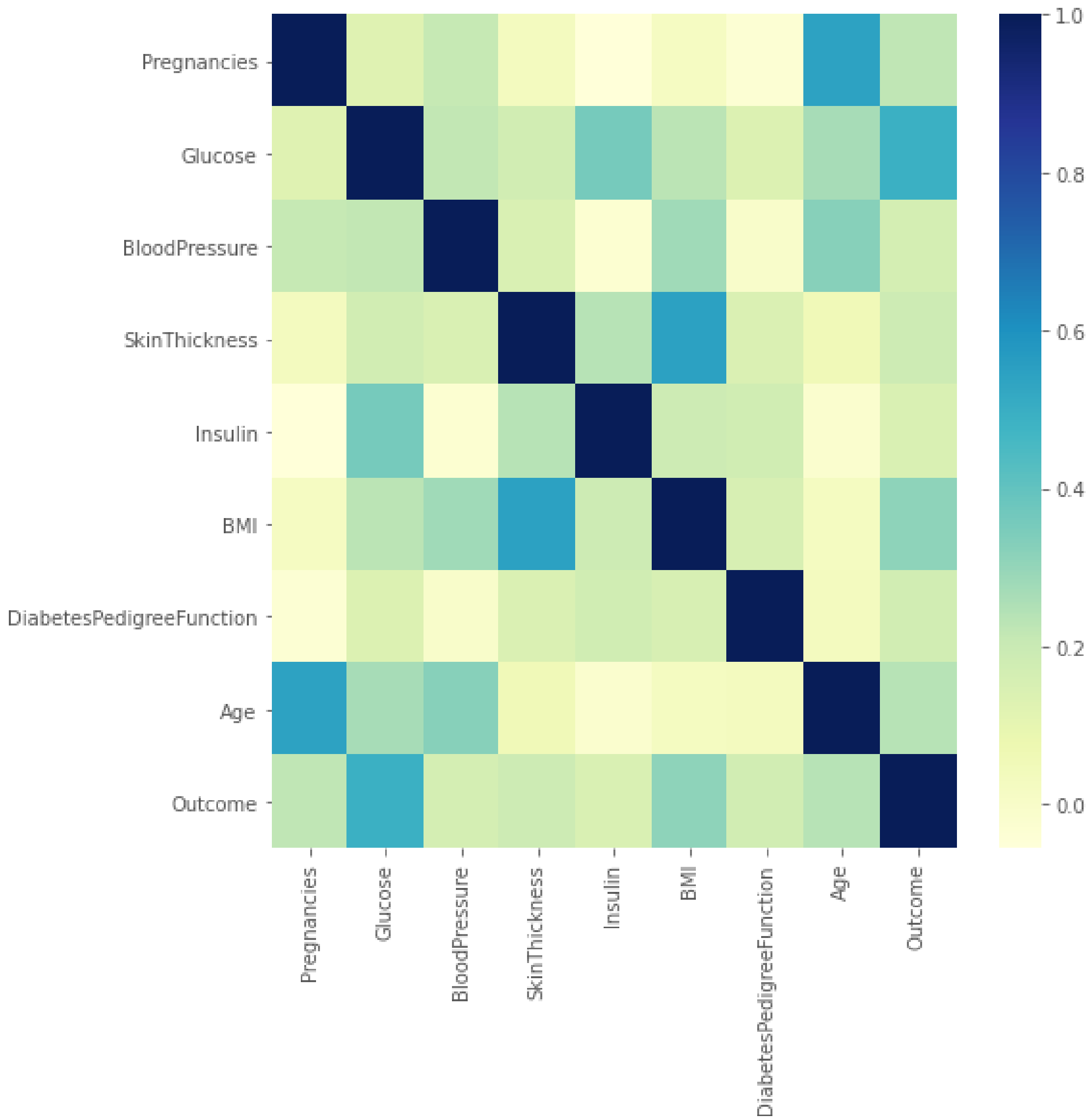
[29]:
```
plt.figure(figsize=(8,8))
sns.heatmap(Dia_Data.corr(),cmap='YlGnBu')
Dia_Data.corr()
```

[29]:
```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.128213       0.208615       0.032568
Glucose                      0.128213  1.000000       0.218937       0.172143
BloodPressure                0.208615  0.218937       1.000000       0.147809
SkinThickness                0.032568  0.172143       0.147809       1.000000
Insulin                     -0.055697  0.357573      -0.028721       0.238188
BMI                          0.021546  0.231400       0.281132       0.546951
DiabetesPedigreeFunction    -0.033523  0.137327      -0.002378       0.142977
Age                          0.544341  0.266909       0.324915       0.054514
Outcome                      0.221898  0.492782       0.165723       0.189065

                           Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies              -0.055697  0.021546                 -0.033523
Glucose                   0.357573  0.231400                  0.137327
BloodPressure            -0.028721  0.281132                 -0.002378
SkinThickness             0.238188  0.546951                  0.142977
Insulin                   1.000000  0.189022                  0.178029
BMI                       0.189022  1.000000                  0.153506
DiabetesPedigreeFunction  0.178029  0.153506                  1.000000
```

```
Age                        -0.015413  0.025744                  0.033561
Outcome                     0.148457  0.312249                  0.173844

                                 Age   Outcome
Pregnancies                 0.544341  0.221898
Glucose                     0.266909  0.492782
BloodPressure               0.324915  0.165723
SkinThickness               0.054514  0.189065
Insulin                    -0.015413  0.148457
BMI                         0.025744  0.312249
DiabetesPedigreeFunction    0.033561  0.173844
Age                         1.000000  0.238356
Outcome                     0.238356  1.000000
```



[30]:
```
#The Above Chart shows Glucose, BMI, and age are strongly related with the␣
↪outcome
```

Project Task: Week 3

[31]:
```
#Devise strategies for model building. It is important to decide the right␣
↪validation framework.
#Express your thought process.
```

Since target has only two catgeories ,We will use Classification rather than regression . We will be using Random forest classification method as it is based on ensamble learning model and provides more accuarte results than single decision tree. We will use train-test-split .

[32]:
```
#Apply an appropriate classification algorithm to build a model.
#Compare various models with the results from KNN algorithm.
```

[33]:
```
#Importing Important Libraries to Build the model
```

[34]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier


from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
```

[35]:
```
Data_input = Dia_Data[['Pregnancies','Glucose','BloodPressure',␣
↪'SkinThickness','Insulin','BMI',
                        'DiabetesPedigreeFunction','Age']]
Data_output = Dia_Data['Outcome']
Data_output = np.ravel(Data_output)
```

[36]:
```
Data_input, Data_output
```

[36]:
```
(     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
 0              6      148             72             35     30.5  33.6
 1              1       85             66             29     30.5  26.6
 2              8      183             64             23     30.5  23.3
 3              1       89             66             23     94.0  28.1
 4              0      137             40             35    168.0  43.1
 ..           …        …             …             …      …   …
 763           10      101             76             48    180.0  32.9
 764            2      122             70             27     30.5  36.8
 765            5      121             72             23    112.0  26.2
 766            1      126             60             23     30.5  30.1
```

```
767            1       93             70             31     30.5  30.4

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

[768 rows x 8 columns],
array([1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
```

```
        0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
        0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0,
        0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
        1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0]))
```

[37]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Data_input,Data_output,␣
 ↪random_state =111)
```

[38]:
```
#initialising the classifiers
```

[39]:
```
lr_classifier =LogisticRegression(solver='lbfgs',max_iter=1000)
rf_classifer  =RandomForestClassifier(n_estimators =10)
```

[40]:
```
#using Logistic regression for model making
```

[41]:
```
lr_classifier.fit(X_train,y_train)
```

[41]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[42]:
```
y_lr_predict= lr_classifier.predict(X_test)
```

[43]:
```
y_lr_predict
```

[43]:
```
array([0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0])
```

[44]:
```
print('RSE\n',np.sqrt(metrics.mean_squared_error(y_lr_predict,y_test)),'\n')
print('Classification report\n')
print(classification_report(y_test,y_lr_predict))
print('accuracy\n',metrics.accuracy_score(y_test,y_lr_predict),'\n')
print('Confusion Matrix\n',confusion_matrix(y_test,y_lr_predict))
```

```
RSE
 0.48947250518628044
```

```
Classification report

              precision    recall  f1-score   support

           0       0.81      0.83      0.82       126
           1       0.66      0.62      0.64        66

    accuracy                           0.76       192
   macro avg       0.73      0.73      0.73       192
weighted avg       0.76      0.76      0.76       192

accuracy
 0.7604166666666666

Confusion Matrix
 [[105  21]
 [ 25  41]]
```

[45]:
```
#Thus the above shows the results from logistic regression
```

[46]:
```
#using random forest now
```

[47]:
```
rf_classifer.fit(X_train,y_train)
```

[47]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[48]:
```
y_rf_predict = rf_classifer.predict(X_test)
```

[49]:
```
y_rf_predict
```

[49]:
```
array([0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0])
```

[50]:
```
print('RSE\n',np.sqrt(metrics.mean_squared_error(y_rf_predict,y_test)),'\n')
print('Classification report\n')
print(classification_report(y_test,y_rf_predict))
print('accuracy\n',metrics.accuracy_score(y_test,y_rf_predict),'\n')
print('Confusion Matrix\n',confusion_matrix(y_test,y_rf_predict))
```

```
RSE
 0.5448623679425842

Classification report

              precision    recall  f1-score   support

           0       0.76      0.81      0.78       126
           1       0.58      0.50      0.54        66

    accuracy                           0.70       192
   macro avg       0.67      0.65      0.66       192
weighted avg       0.69      0.70      0.70       192

accuracy
 0.703125

Confusion Matrix
 [[102  24]
 [ 33  33]]
```

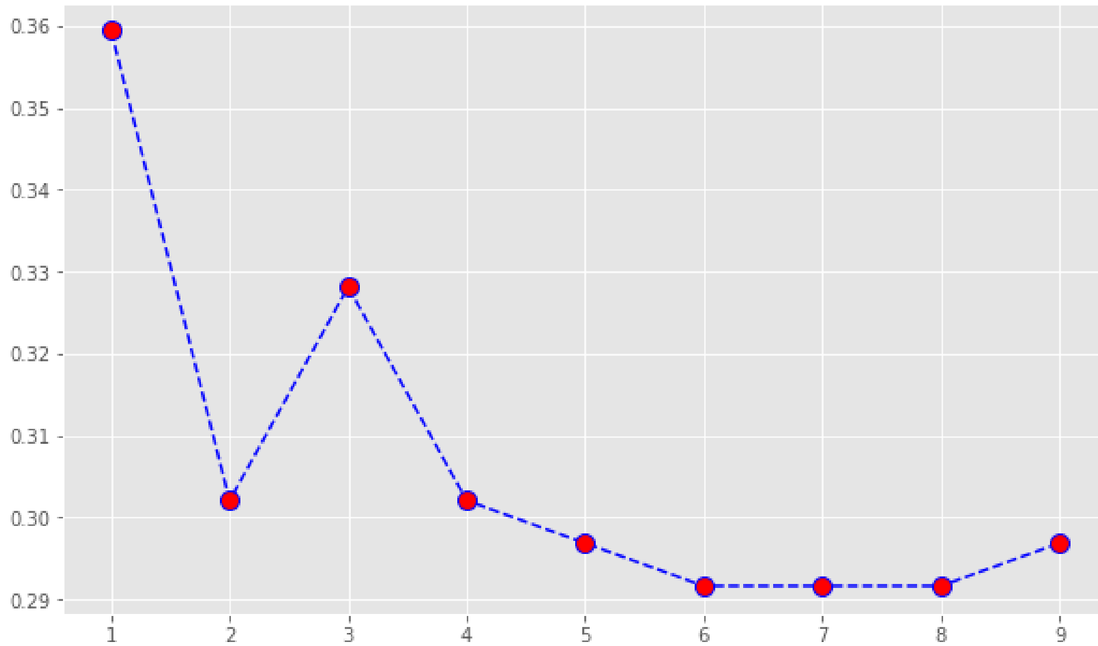
[51]:
```
#the above show the results for Random Forest
```

[52]:
```
#using Knn
```

[53]:
```
error_rate = []
for i in range(1,10):
    Knn= KNeighborsClassifier(n_neighbors=i)
    Knn.fit(X_train,y_train)
    pred_i = Knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

[54]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,10),error_rate,color='blue', linestyle='dashed',␣
 ↪marker='o',markerfacecolor='red',markersize=10)
```

[54]:
```
[<matplotlib.lines.Line2D at 0x7f4eb998fd50>]
```



[55]:
```
# since the error is minimum at K = 7 , we will use 7 as value for no of␣
↪neighbours
```

[56]:
```
knn_classifier =KNeighborsClassifier(n_neighbors=7)
```

[57]:
```
knn_classifier.fit(X_train,y_train)
```

[57]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=7, p=2,
                     weights='uniform')
```

[58]:
```
y_knn_predict = knn_classifier.predict(X_test)
```

[59]:
```
print('RSE\n',np.sqrt(metrics.mean_squared_error(y_knn_predict,y_test)),'\n')
print('Classification report\n')
print(classification_report(y_test,y_knn_predict))
print('accuracy\n',metrics.accuracy_score(y_test,y_knn_predict),'\n')
print('Confusion Matrix\n',confusion_matrix(y_test,y_knn_predict))
```

```
RSE
 0.5400617248673217

Classification report

              precision    recall  f1-score   support
```

```
              0       0.80      0.75      0.77       126
              1       0.57      0.64      0.60        66

       accuracy                           0.71       192
      macro avg       0.68      0.69      0.69       192
   weighted avg       0.72      0.71      0.71       192

accuracy
 0.7083333333333334

Confusion Matrix
 [[94 32]
 [24 42]]
```

Project Week 4

[60]:
```
#Create a classification report by analyzing sensitivity, specificity, AUC (ROC␣
 ↪curve), etc.
#Please be descriptive to explain what values of these parameter you have used.
```

Classfication report for the three mdoels used are given below

1. for Logistic regression

RSE= 0.48947250518628044 , accuracy = 0.7604166666666666 Classification report

precision recall f1-score support

0 0.81 0.83 0.82 126 1 0.66 0.62 0.64 66

accuracy 0.76 192 macro avg 0.73 0.73 0.73 192 weighted avg 0.76 0.76 0.76 192

Confusion Matrix [[105 21] [25 41]]

2.for random forest

RSE=0.535218024111047, accuracy= 0.7135416666666666 Classification report

precision recall f1-score support

0 0.76 0.83 0.79 126 1 0.60 0.48 0.54 66

accuracy 0.71 192 macro avg 0.68 0.66 0.67 192 weighted avg 0.70 0.71 0.70 192

Confusion Matrix [[105 21] [34 32]]

For KNN RSE= 0.5400617248673217 ,accuracy=0.7083333333333334 Classification report

precision recall f1-score support

0 0.80 0.75 0.77 126 1 0.57 0.64 0.60 66

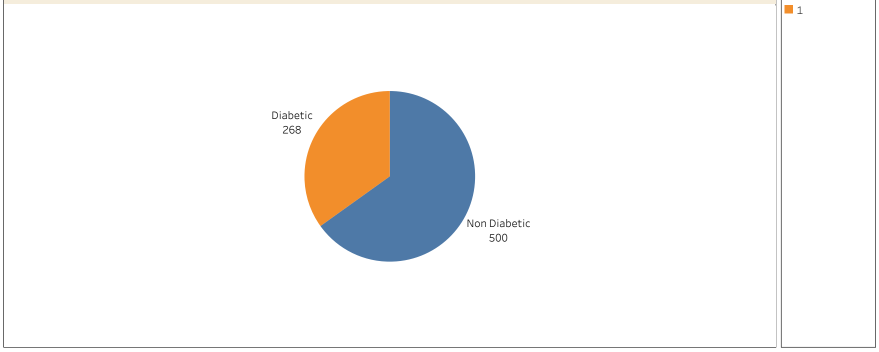
accuracy 0.71 192 macro avg 0.68 0.69 0.69 192 weighted avg 0.72 0.71 0.71 192

Confusion Matrix [[94 32] [24 42]]

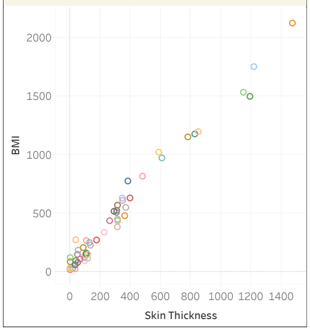
[62]:
```
# Please find the detailed classifcation report above .And teh alogorithm has␣
↪used the default threshold as 0.5.
```

Data Reporting and Dash board creation is given below

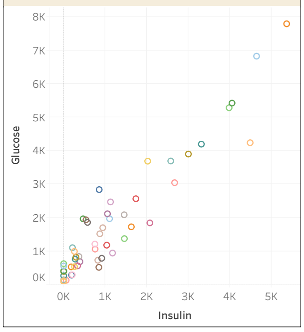
Pie chart to describe the diabetic or non-diabetic population

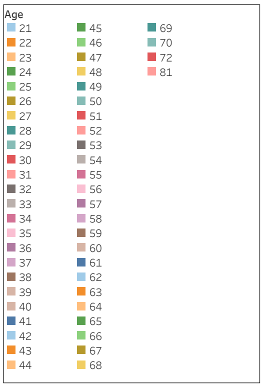


Scatter Chart between BMI & Skin Thickness

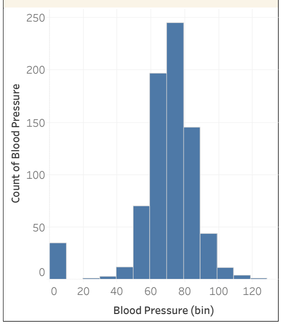


Scatter chart between Glucose and Insulin

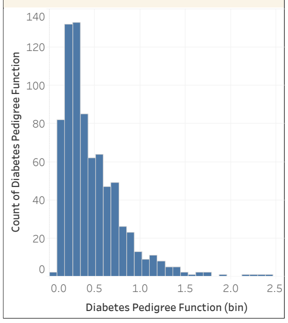




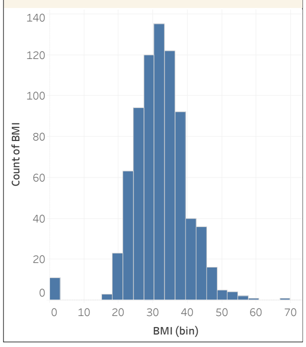
Blood Pressure Data Distribution



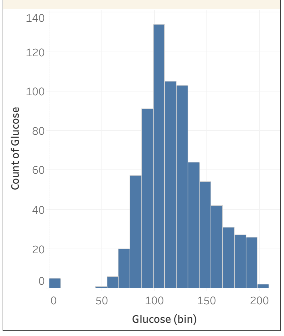
DiabetesPedigreeFunction Data Distribution



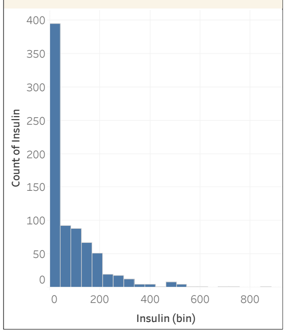
BMI Data Distribution



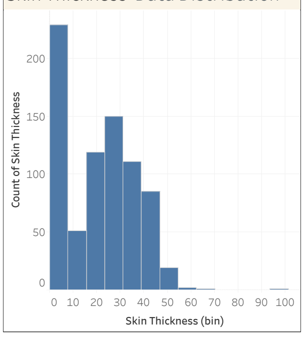
Glucose Data Distribution



Insulin Data Distribution



Skin Thickness Data Distribution

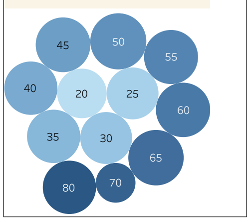


Heatmap between Age & Outcome

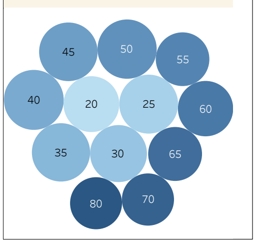
PCC 0.98422 – 1

Outco..	20	25	30	35	40	45	50	55	60	65	70	80
0	0.99837	0.99498	0.99599	0.99500	0.98432	0.99630	0.99367	0.99975	0.99872	0.99437		
1	0.99758	0.99536	0.99980	0.98422	0.99802	0.99758	0.99720	0.99470	0.99994	1.00000		

Bubble chart for Blood Pressure



Bubble chart for BMI



Bubble chart for Glucose