Project1

March 22, 2020

1 Project 1

2 Linear Regression

3 1) Explore Data

[56]:

```
import warnings
warnings.filterwarnings('ignore')


import pandas as pd
import numpy as np
from plotnine import *

import statsmodels.api as sm


from sklearn.linear_model import LogisticRegression # Logistic Regression Model
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.metrics import r2_score, mean_squared_error

from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.model_selection import cross_val_score # cross validation metrics
from sklearn.model_selection import cross_val_predict # cross validation metrics
```

[57]:

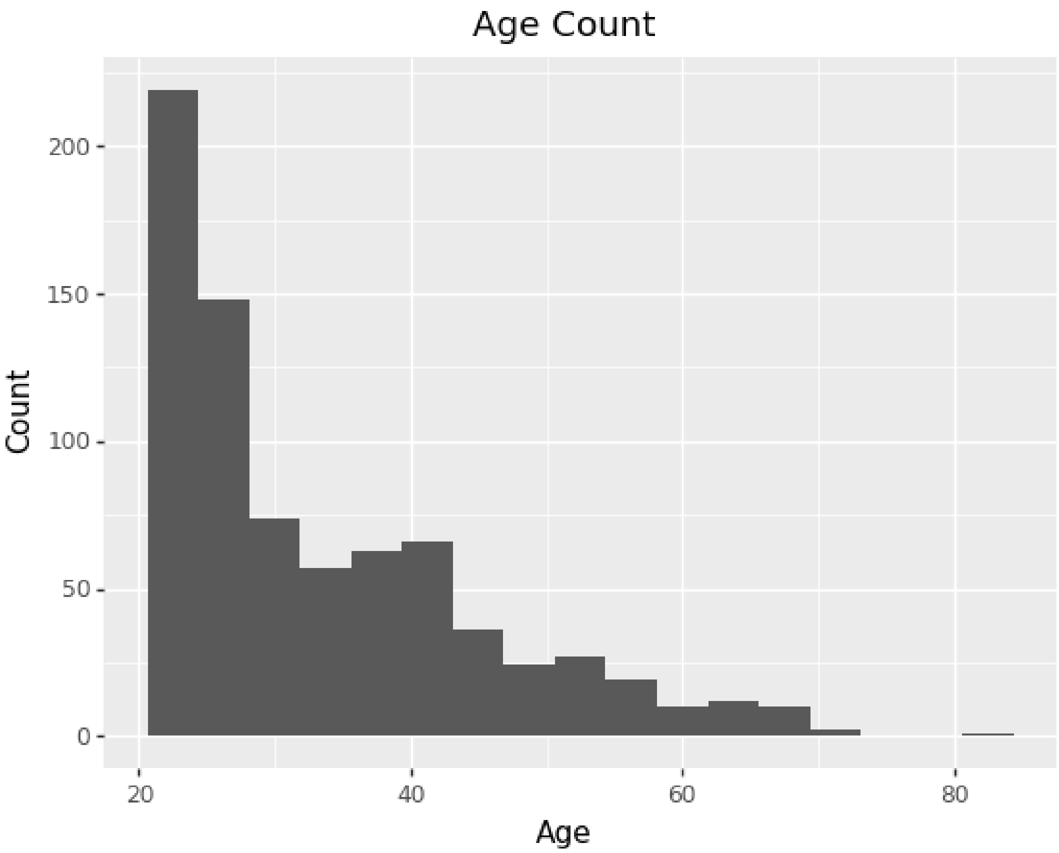
```
data = "https://raw.githubusercontent.com/cmparlettpelleriti/
 ↪CPSC392ParlettPelleriti/master/Data/diabetes2.csv"
```

[52]:

```
#grabs data from online link
diabetes = pd.read_csv(data)
#reads data from csv file
diabetes.head()
#looks at the first 5 observations of the csv file
```

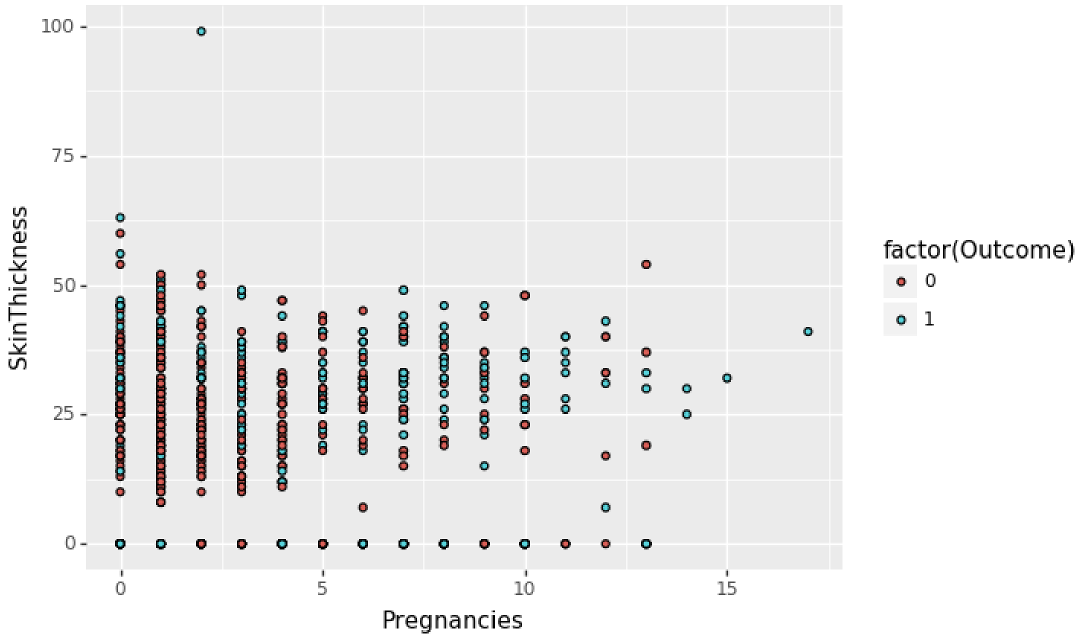
```
[52]:    Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
      0            6      148             72             35        0  33.6
      1            1       85             66             29        0  26.6
      2            8      183             64              0        0  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
```

```
[4]: (ggplot(diabetes, aes("Age"))
      + geom_histogram(aes(fill = "Age"))
      + labs(title = "Age Count", x = "Age", y = "Count"))
```



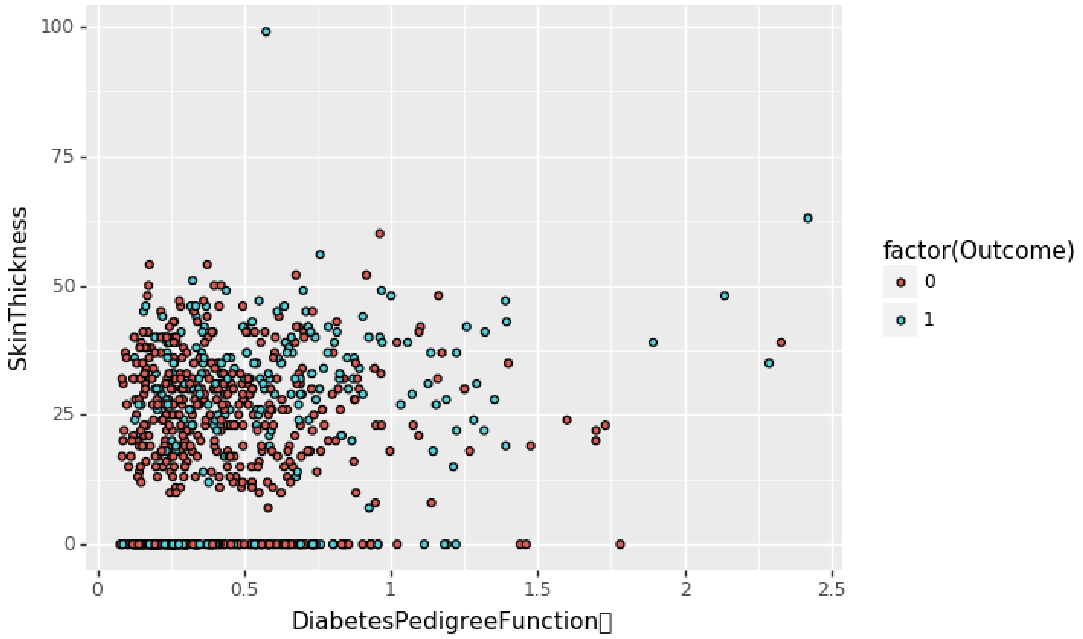
```
[4]: <ggplot: (301571917)>
```

[5]:
```
(ggplot(diabetes, aes("Pregnancies", "SkinThickness"))
 + geom_point(aes(fill = "factor(Outcome)")))
```



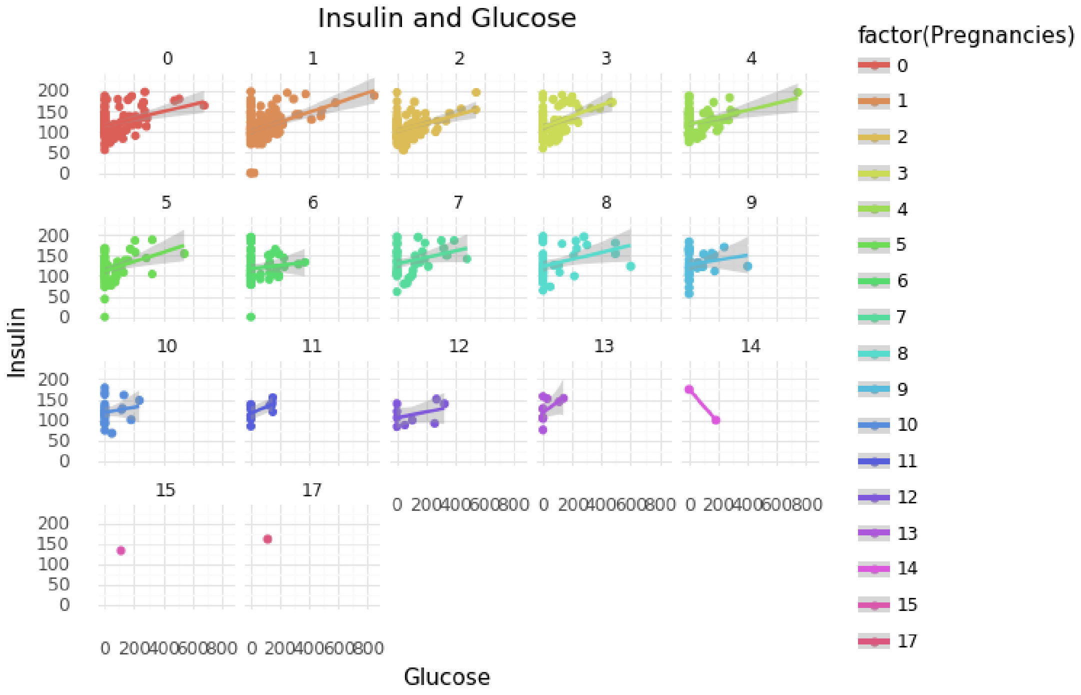
[5]:
```
<ggplot: (301578353)>
```

[6]:
```
(ggplot(diabetes, aes("DiabetesPedigreeFunction         ", "SkinThickness"))
 + geom_point(aes(fill = "factor(Outcome)")))
```



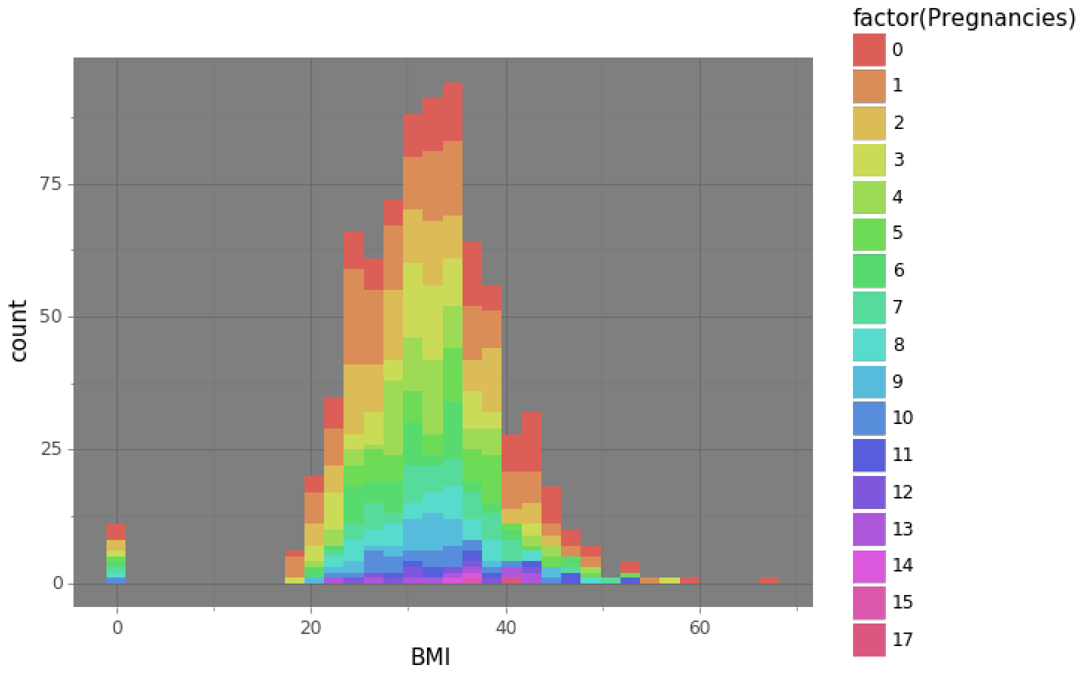
```
[6]: <ggplot: (301571853)>
```

```
[7]: (ggplot(diabetes, aes('Insulin', 'Glucose', color = 'factor(Pregnancies)'))
      + geom_point() #creates point graph
      + stat_smooth(method='lm') #smooths the slope line
      + facet_wrap('~Pregnancies') #seperates each mfr into its own graph
      + theme_minimal()
      + labs(title = "Insulin and Glucose",
         x = "Glucose", y = "Insulin")) #labels the graph
```



```
[7]: <ggplot: (302875505)>
```

```
[8]: (ggplot(diabetes, aes('BMI'))
      + theme_dark()
      + geom_histogram(aes(fill = "factor(Pregnancies)")))
```



[8]: `<ggplot: (302919769)>`

4 2) Building My Model

[9]:
```python
diabetes.head()
```

[9]:
```
   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
```

[10]:
```python
# creates predictors
predictors = ["Glucose", "BloodPressure", "Insulin", "Age", "SkinThickness",␣
 ↪"Pregnancies", "DiabetesPedigreeFunction"]
```

```
#creates test data and training data
X_train, X_test, y_train, y_test = train_test_split(diabetes[predictors],␣
 ↪diabetes["BMI"], test_size=0.2)
```

[11]:
```
#Standardization
zscore = StandardScaler()
zscore.fit(X_train)
Xz_train = zscore.transform(X_train)
Xz_test = zscore.transform(X_test)
```

5 3) Evaluate your model

[12]:
```
# create linearRegression model
LR_Model = LinearRegression()
```

[13]:
```
# fit logModel
LR_Model.fit(Xz_train,y_train)
```

[13]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

[17]:
```
BMI_preds = LR_Model.predict(Xz_test)
```

[18]:
```
mean_squared_error(y_test,BMI_preds)
```

[18]:
```
47.737293995292326
```

[19]:
```
r2_score(y_test,BMI_preds)
```

[19]:
```
0.23325950918080884
```

[19]:
```
# ERROR
# My model did not do to well. My mean-squared error was 47.73

# ACCURACY
# Also my model had a r score of 0.23 which is not very good.
```

6 4) Interpret the coefficients to your model

[20]:
```
coefficients = pd.DataFrame({"Coef":LR_Model.coef_,
              "Name": predictors})
coefficients = coefficients.append({"Coef": LR_Model.intercept_,
                    "Name": "intercept"}, ignore_index = True)
```

[21]:
```
coefficients
```

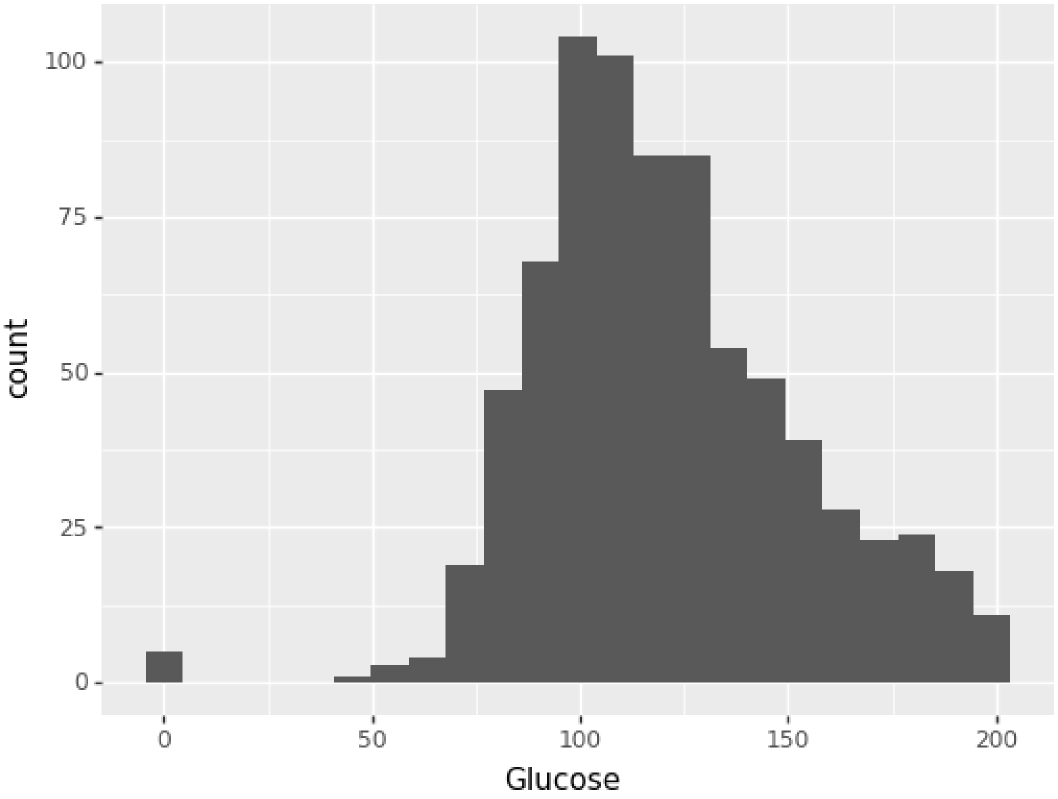
[21]:
```
          Coef                      Name
0   1.546936                   Glucose
1   1.120048             BloodPressure
2  -0.492824                   Insulin
3  -0.227689                       Age
4   3.038312             SkinThickness
5   0.017818               Pregnancies
6   0.377344  DiabetesPedigreeFunction
7  31.975407                 intercept
```

[23]:
```
# These coefficients show the affect of these variables on the BMI level
# For one stdv increase of Glucose there is a 1.547 increase in BMI
# For one stdv increase of BloodPressure there is a 1.120 increase in BMI
# For one stdv increase of Insulin there is a -0.493 decrease in BMI
# For one stdv increase of Age there is a -0.228 decrease in BMI
# For one stdv increase of SkinThickness there is a 3.038 increase in BMI
# For one stdv increase of Pregnancies there is a 0.018 in BMI
# For one stdv increase of DiabetesPedigreeFunction there is a 0.377 increaes␣
 ↪in BMI
# If there were no variables involved, y-intercept is 31.975
```

7 Logistic Regression

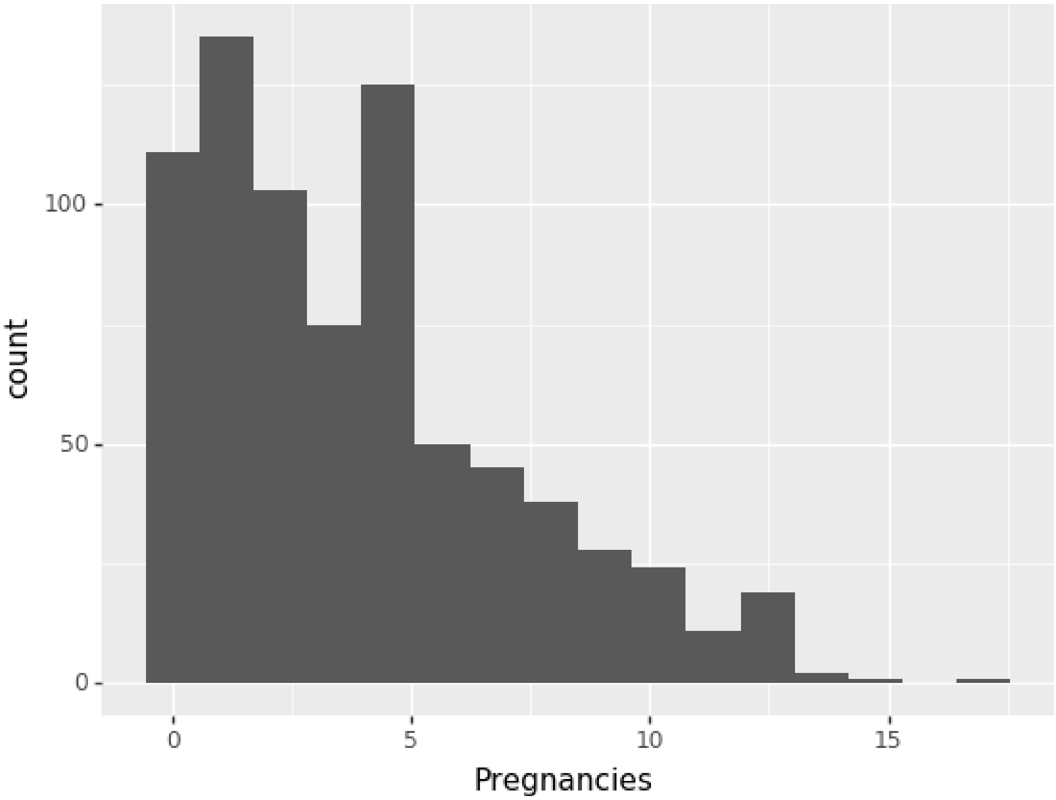
8 1) Explore Data

[49]:
```
(ggplot(diabetes, aes("Glucose"))
 + geom_histogram())
```



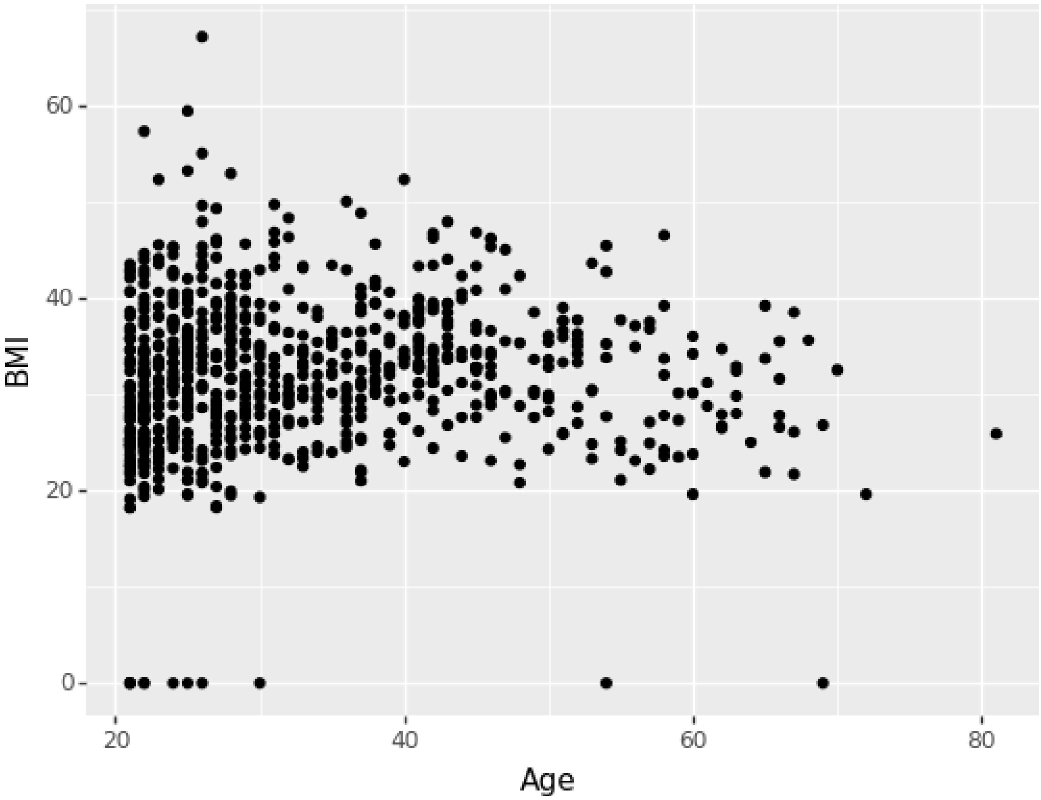
```
[49]: <ggplot: (317381793)>
```

```
[46]: (ggplot(diabetes, aes("Pregnancies"))
       + geom_histogram())
```



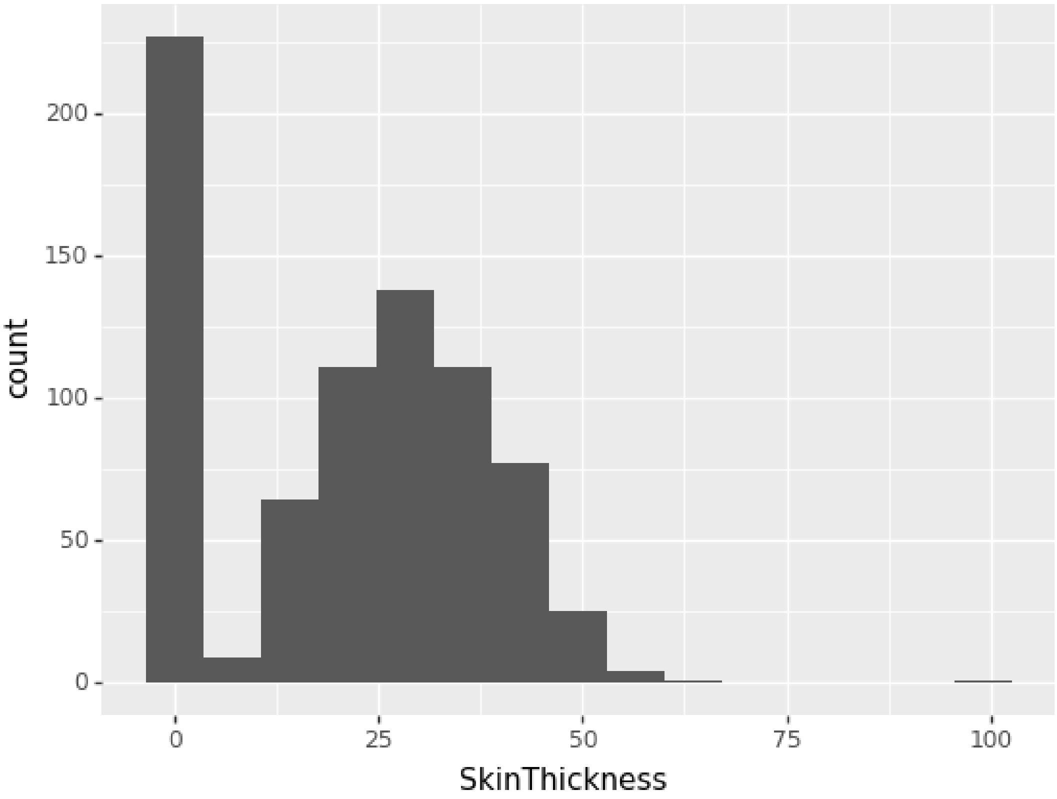
```
[46]: <ggplot: (317578725)>
```

```
[53]: (ggplot(diabetes, aes("Age", "BMI"))
       + geom_point())
```



```
[53]: <ggplot: (317043029)>
```

```
[54]: (ggplot(diabetes, aes("SkinThickness"))
       + geom_histogram())
```



```
[54]: <ggplot: (316943769)>
```

9 2) Building My Model

```
[53]: diabetes.head()
```

```
[53]:    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
```

[63]:

```
# Kfold

X = diabetes[["Glucose", "Pregnancies", "BloodPressure", "SkinThickness",␣
 ↪"Insulin", "BMI","DiabetesPedigreeFunction","Age"]]
y = diabetes["Outcome"]

# create k-fold object
kf = KFold(n_splits = 8)
kf.split(X)

# standardization
zScore = StandardScaler()
zScore.fit(X)
Xz = zScore.transform(X)

lr = LogisticRegression() #create model

acc = [] #create empty list to store accuracy for each fold
```

[64]:

```
# Use a for loop to loop through each fold and train a model, then add the␣
 ↪accuracy to acc.

for train_indices, test_indices in kf.split(Xz):
    # Get your train/test for this fold
    X_train_k = X.iloc[train_indices]
    X_test_k  = X.iloc[test_indices]
    y_train_k = y[train_indices]
    y_test_k  = y[test_indices]

    # model
    model = lr.fit(X_train_k, y_train_k)
    # record accuracy
    acc.append(accuracy_score(y_test_k, model.predict(X_test_k)))

#print overall acc
print(acc)
np.mean(acc)
```

```
[0.7395833333333334, 0.8125, 0.7083333333333334, 0.75, 0.7708333333333334,
0.8125, 0.78125, 0.7916666666666666]
```

[64]:

```
0.7708333333333334
```

10 Evaluating my Model

```
[66]: len(diabetes)
```

```
[66]: 768
```

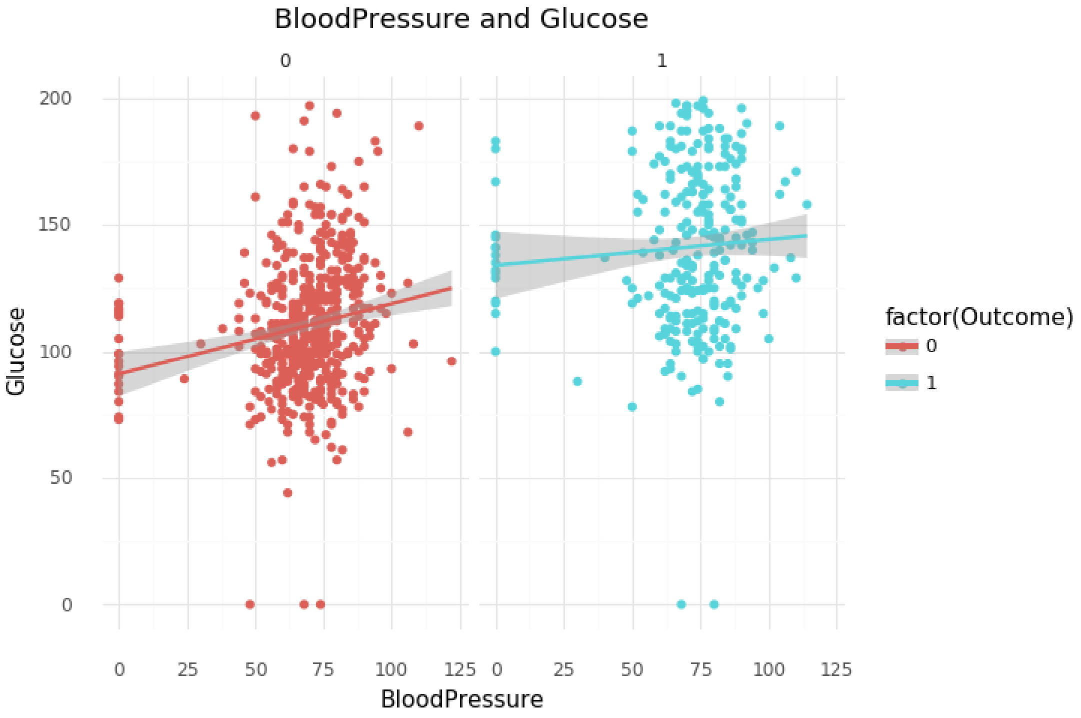
Metrics/Interpretation of model: I used the accuracy score to determine how accurate my model was. I got an accuracy score of 0.77 which is a pretty accurate model score so I am happy with my model

Justification for Cross-val technique: Since there are hundreds of observations in this dataset(768), I chose to do K fold as my cross-validation technique because it is less computationally expensive.

11 Data Viz

12 #1

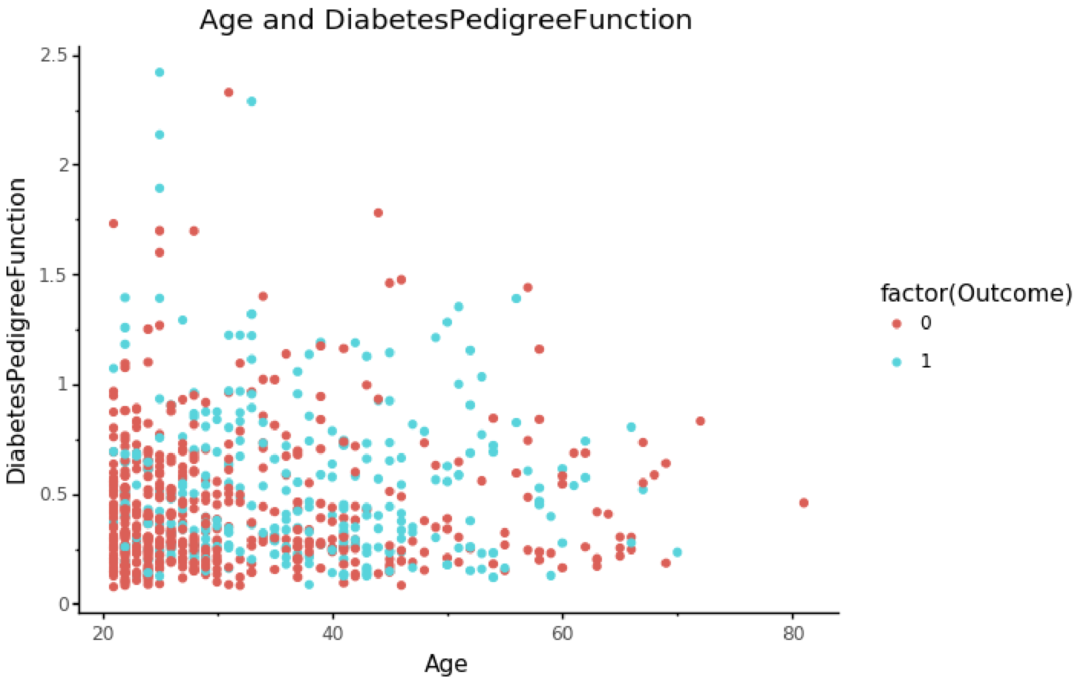
```
[37]: (ggplot(diabetes, aes('BloodPressure', 'Glucose', color = "factor(Outcome)"))
       + geom_point() #creates point graph
       + stat_smooth(method='lm') #smooths the slope line
       + facet_wrap('~Outcome') #seperates each mfr into its own graph
       + theme_minimal()
       + labs(title = "BloodPressure and Glucose",
           x = "BloodPressure", y = "Glucose")) #labels the graph
```



```
[37]: <ggplot: (317131621)>
```

13 #2

```
[59]: (ggplot(diabetes, aes('Age', 'DiabetesPedigreeFunction', color =␣
       ↪'factor(Outcome)'))
       + geom_point() #creates point graph
       + theme_classic()
       + labs(title = "Age and DiabetesPedigreeFunction",
           x = "Age", y = "DiabetesPedigreeFunction")) #labels the graph
```



```
[59]: <ggplot: (317195853)>
```

```
[ ]:
```