Project 1

Use the data *diabetes2.csv* for this project. More information about the dataset can be found here: https://www.kaggle.com/kandij/diabetes-dataset (https://www.kaggle.com/kandij/diabetes-dataset)

Linear Regression

Can you predict BMI based on other features in the dataset?

1. Explore the Data
2. Build your Model
    - Build a Linear Regression Model using train_test_split() for your cross-validation
    - Standardize your continuous predictors
3. Evaluate your model
    - How did your model do? What metrics do you use to support this?
4. Interpret the coefficients to your model
    - In the context of this problem, what do the coefficients represent?

1. Explore the Data

In [86]:
```
import warnings
warnings.filterwarnings('ignore')

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


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

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
```

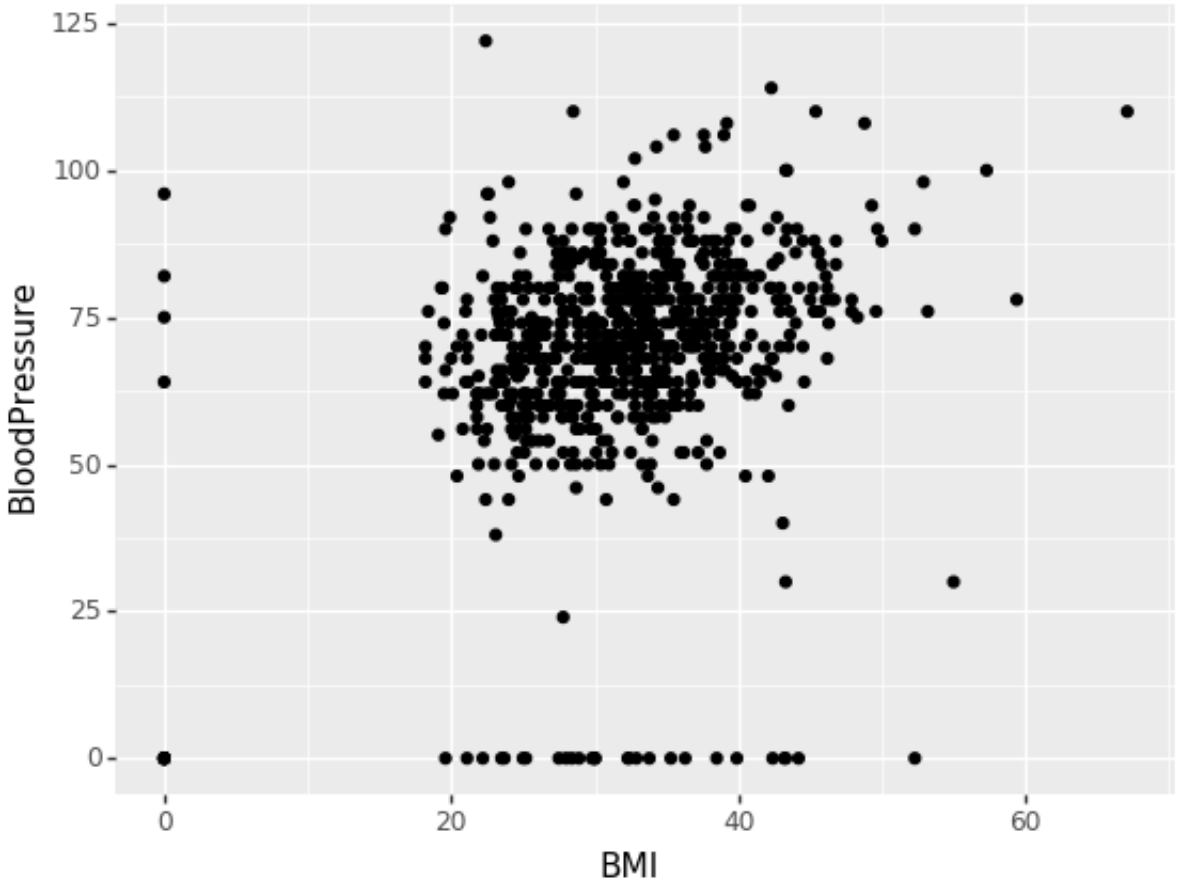
In [87]:
```
aye = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392ParlettPelleriti/master/Data/diabetes2.csv")

aye.head()
```

Out[87]:

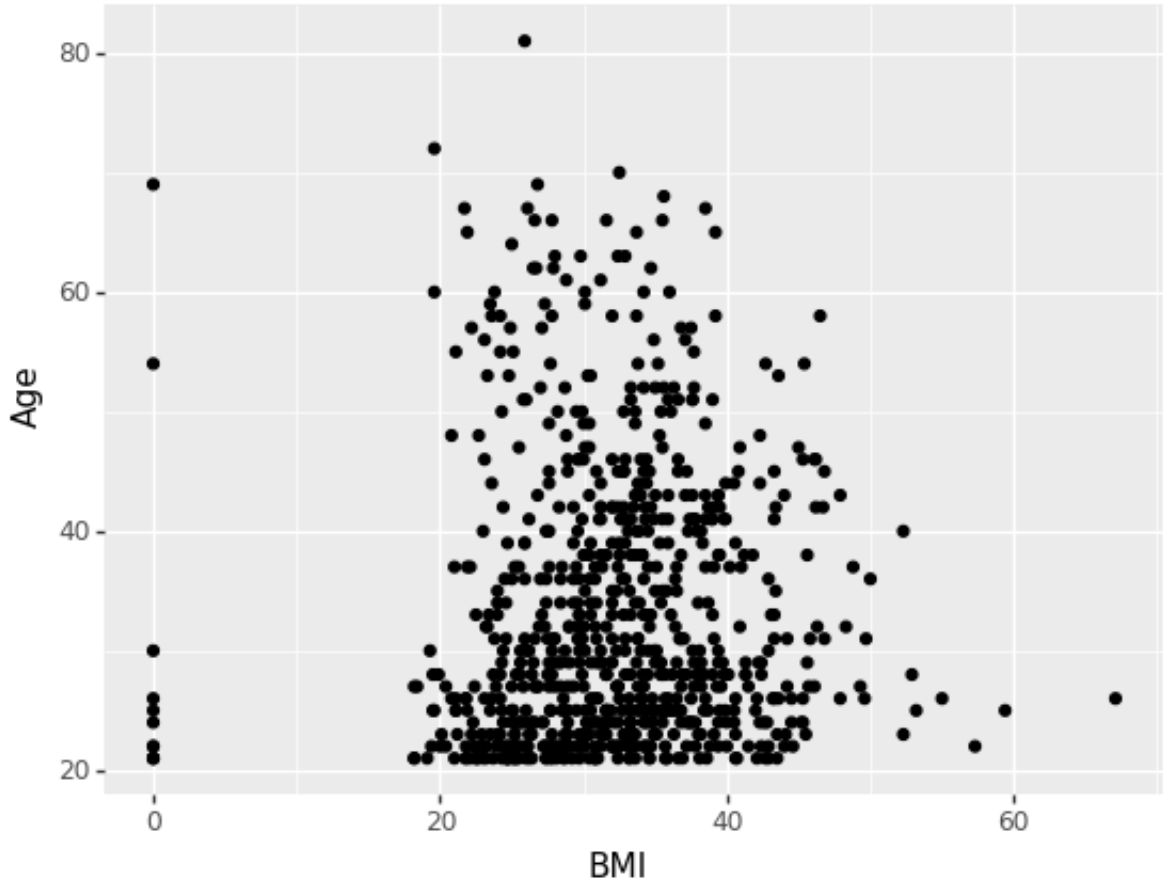
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [25]:
```
(ggplot(aye, aes(x = "BMI", y = "BloodPressure")) + geom_point())
```



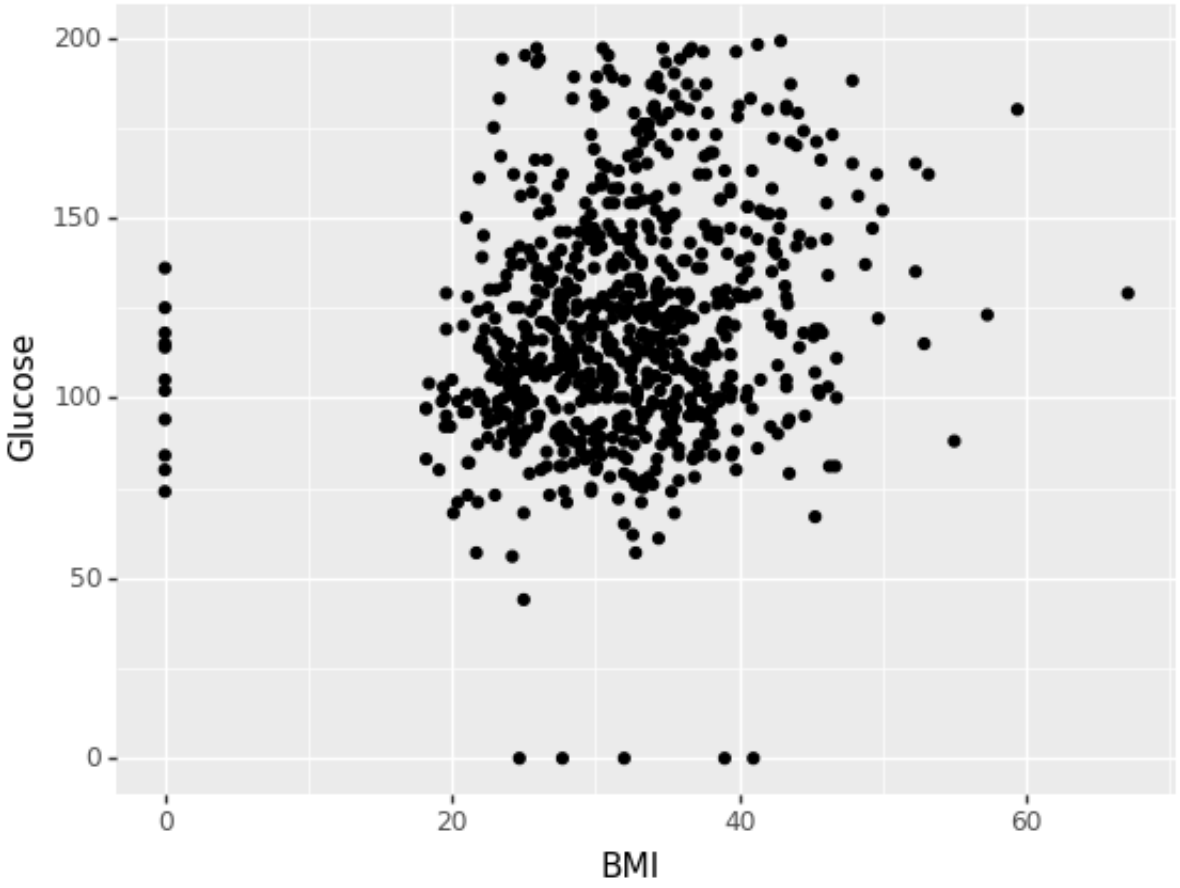
Out[25]: <ggplot: (309083685)>

In [26]:
```
(ggplot(aye, aes(x = "BMI", y = "Age")) + geom_point())
```



Out[26]: <ggplot: (309177261)>

In [27]:
```
(ggplot(aye, aes(x = "BMI", y = "Glucose")) + geom_point())
```



Out[27]: <ggplot: (309299525)>

2. Build Your Model

In [76]:
```
predictors = ["Glucose", "BloodPressure", "SkinThickness", "Insulin",
"Outcome"]

X_train, X_test, y_train, y_test = train_test_split(aye[predictors], a
ye["BMI"], test_size=0.2)
```

In [77]:
```
X_train.shape
```

Out[77]: (614, 5)

In [78]:
```
X_test.shape
```

Out[78]: (154, 5)

In [79]:
```
X_train.head()
```

Out[79]:

	Glucose	BloodPressure	SkinThickness	Insulin	Outcome
377	87	60	37	75	0
577	118	80	0	0	1
733	106	56	27	165	0
136	100	70	26	50	0
487	173	78	32	265	0

In [80]:
```
#standardization
zscore = StandardScaler()
zscore.fit(X_train)
Xz_train = zscore.transform(X_train)
Xz_test = zscore.transform(X_test)
```

In [81]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[81]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, norma
lize=False)
```

In [82]:
```
# predictions
y_pred = model.predict(Xz_train)
y_pred[1:10]
```

Out[82]:
```
array([24.94721331, 16.85776493, 16.90356397, 17.03515384, 25.105811
95,
       25.25321535, 17.05710194, 16.64455825, 16.95197945])
```

In [91]:
```
mean_squared_error(y_train, y_pred)
```

Out[91]:
```
204.89127646815243
```

In [92]:
```
r2_score(y_train, y_pred)
```

Out[92]:
```
-2.40988018803808
```

3. Evaluate Your Model

How did your model do? What metrics do you use to support this?

The metrics I used to support my model was the mean squared error and r squared.

My model did not do well. The mean squared error is valued at 204.891. With a large mean square error, it shows that my data is widely dispersed. Therefore, it means that my data is not the preferred choioce because this reflects that my data has a lot of errors. If we were to have a smaller mean squared error, then my data would be a good data.

The r squared is negative 2.410. This shows that the regression line is worse than using the mean value, that is why the r squared is a negative value. This also means that my prediction tends to be less accurate.

4. Interpret the coefficients to your model

In [90]:
```
coefficients = pd.DataFrame({"Coef": model.coef_, "Name": predictors})
coefficients = coefficients.append({"Coef": model.intercept_, "Name":
"Intercept"}, ignore_index = True)
coefficients
```

Out[90]:

	Coef	Name
0	0.015036	Glucose
1	0.080597	BloodPressure
2	0.165317	SkinThickness
3	0.000557	Insulin
4	3.933069	Outcome
5	19.721881	Intercept

In the context of this problem, what do the coefficients represent?

The coefficients for Glucose means that for every increase in 1 unit is associated with an increase in BMI by 0.015.

The coefficients for Blood Pressure means that for every increase in 1 unit is associated with an increase in BMI by 0.081.

The coefficients for Skin Thickness means that for every increase in 1 unit is associated with an increase in BMI by 0.165.

The coefficients for Insulin means that for every increase in 1 unit is associated with an increase in BMI by 0.0006.

The coefficients for Outcome (Diabetes) means that for every increase in 1 unit is associated with an increase in BMI by 3.933.

The Intercept means that with if the coefficients were to equal to 0, then the value of the BMI would still be 19.722.

Logistic Regression

Can you predict Diabetes (Outcome) based on other features in the dataset?

1. Explore the Data (if using different variables from Linear Regression)
2. Build your Model
    - Build a Logistic Regression Model using cross-validation
        - What cross-val method did you choose, why?
    - Standardize your continuous predictors
3. Evaluate your model
    - How did your model do? What metrics do you use to support this?

1. Explore the Data

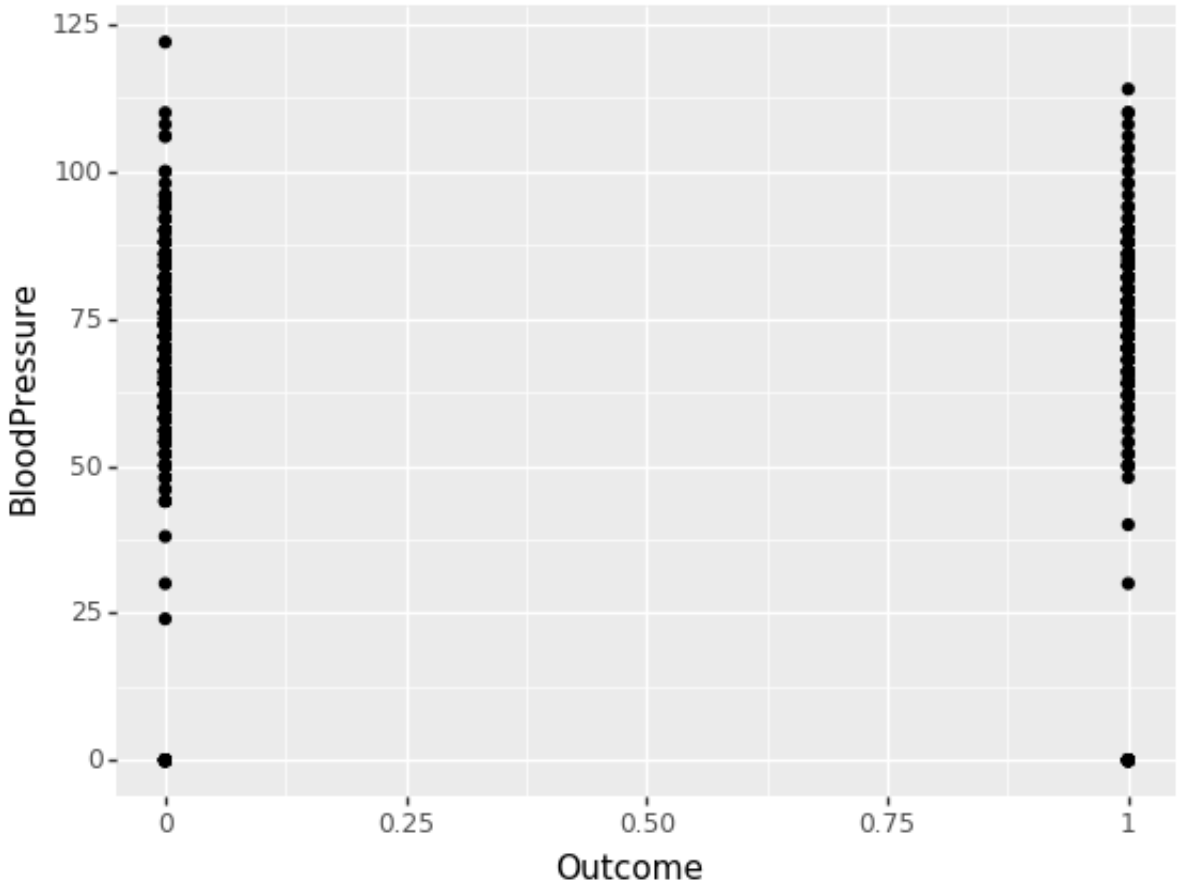
In [89]:
```
aye = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392ParlettPelleriti/master/Data/diabetes2.csv")

aye.head()
```

Out[89]:

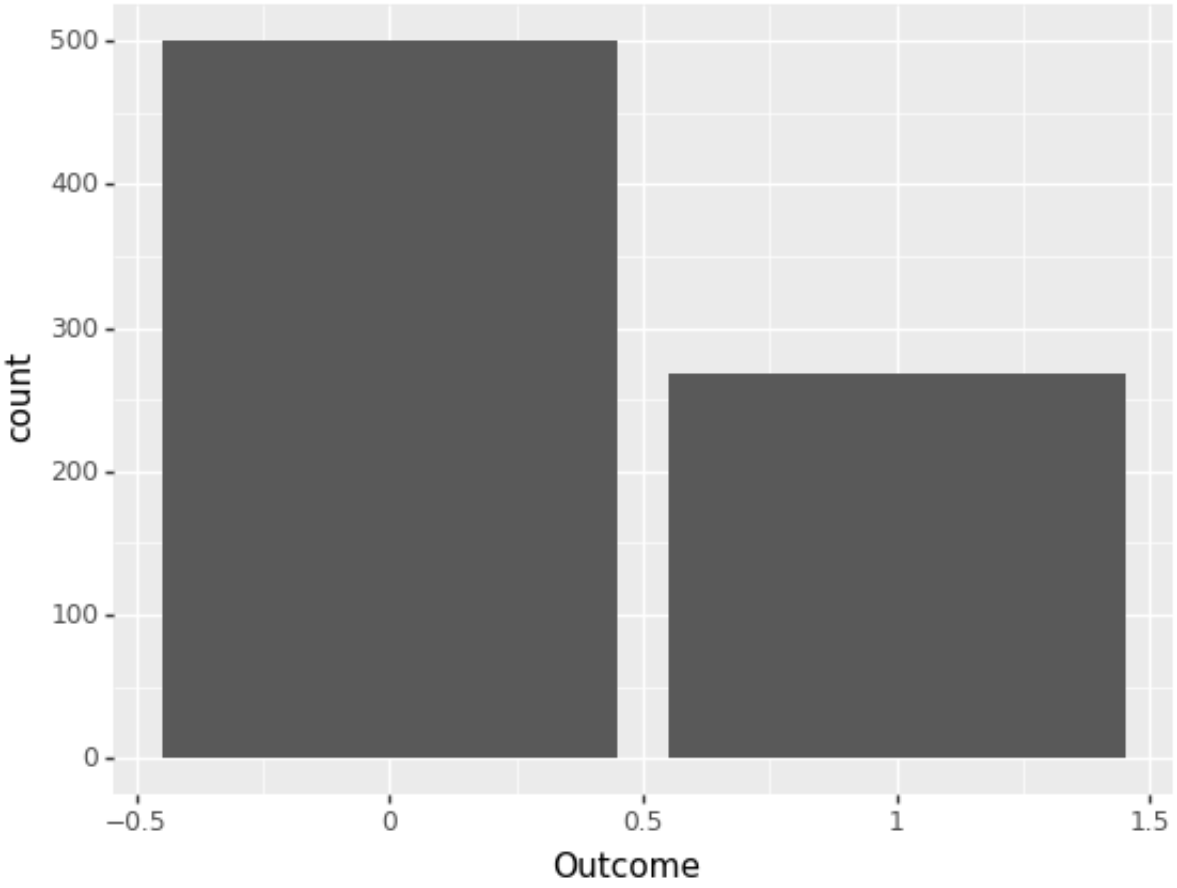
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctic
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [90]:
```
(ggplot(aye, aes(x = "Outcome", y = "BloodPressure")) + geom_point())
```



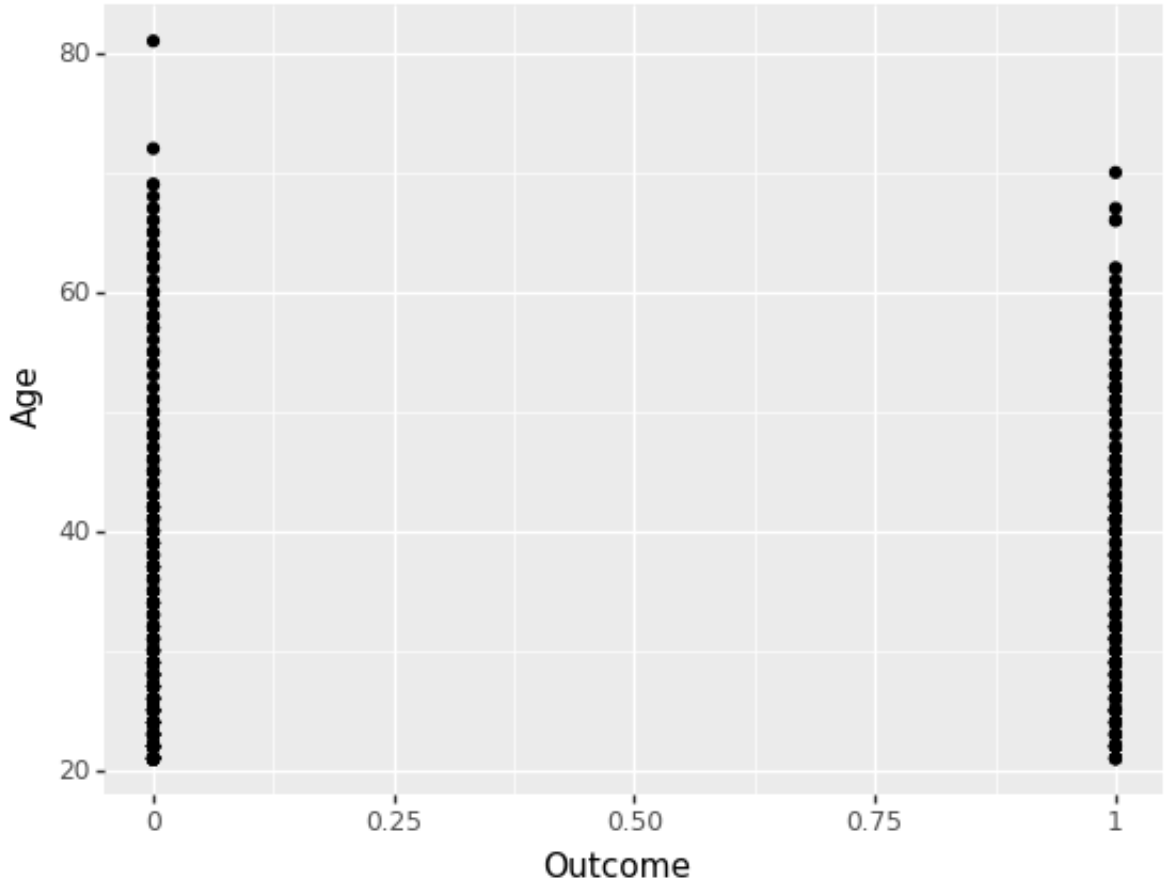
Out[90]: <ggplot: (313401781)>

In [91]:
```
(ggplot(aye, aes(x = "Outcome")) + geom_bar())
```



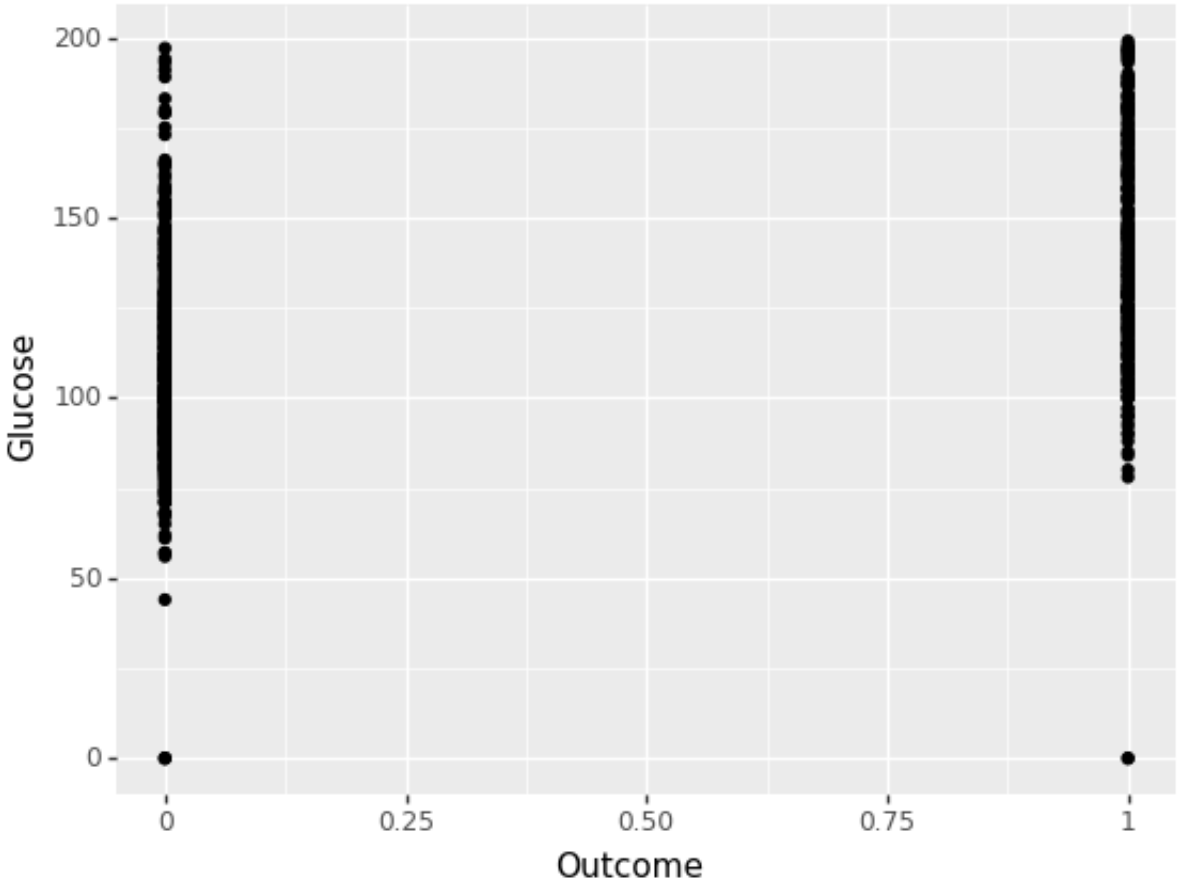
Out[91]: <ggplot: (313354081)>

In [94]:
```
(ggplot(aye, aes(x = "Outcome", y = "Age")) + geom_point())
```



Out[94]: <ggplot: (313355485)>

In [95]:
```
(ggplot(aye, aes(x = "Outcome", y = "Glucose")) + geom_point())
```



Out[95]: <ggplot: (313355273)>

2. Build Your Model

In [96]:
```
X = aye[["Glucose", "BloodPressure", "BMI", "Age"]]
y = aye["Outcome"]

#create LOO
kf = LeaveOneOut()
kf.split(X)

lr = LogisticRegression() #create model

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

In [97]:
```
for train_indices, test_indices in kf.split(X):
    # Get your train/test for this fold
    X_train = X.iloc[train_indices]
    X_test  = X.iloc[test_indices]
    y_train = y[train_indices]
    y_test  = y[test_indices]

    #standardization
    zscore = StandardScaler()
    zscore.fit(X_train)
    Xz_train = zscore.transform(X_train)
    Xz_test = zscore.transform(X_test)

    # model
    model = lr.fit(X_train, y_train)
    # record accuracy
    acc.append(accuracy_score(y_test, model.predict(X_test)))

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

```
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.
0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0
, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0
.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0,
1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0
.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0,
1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1
.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0
.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0
, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1
.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0,
1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0
, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1
.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 1.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0
, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0
```

```
, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0
, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0
, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1
.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1
.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 0
.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0,
0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0
, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0
.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0,
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0
, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1
.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0, 0.0, 1.0]
```

Out[97]: 0.7682291666666666

What cross-val method did you choose, why?

The cross validation method that I used is Leave One Out. I used this method because I do not think that having pregnancies, skin thickness, insulin, diabetes pedigree function, and outcome (diabetes) is relevant to my model. I do not think that pregnancy fits into the category because a pregnant people would be carrying a child inside their stomach that would result in the BMI value to be bigger. Skin thickness is irrelevant because different parts of the body has different "thickness" and does not associate with the model. Insulin does not fit the model because people that take insulin have diabetes. A diabetes pedigree function is a likelihood of diabetes based on family history meaning that we will not know if the person will even have diabetes.

3. Evaluate Your Model

How did your model do? What metrics do you use to support this?

My model did well and was able to predict the model 76.82% correctly. I used the accuracy score to determine how well my model did.

Data Viz

Based on your new understanding of the data create 2 graphs using ggplot/plotnine. These should **not** be graphs you made in the Explore phase of either the Logistic or Linear Regression portion.

Make sure you include at **least** 3 out of these 5 elements in your at least one of your graphs:

1. Custom x-axis labels, y-axis labels and titles
2. Fill and/or Color by a variable
3. Use facet_wrap()
4. Layer multiple geoms
5. Change the theme of your graph (see: https://plotnine.readthedocs.io/en/stable/generated/plotnine.themes.theme.html (https://plotnine.readthedocs.io/en/stable/generated/plotnine.themes.theme.html))

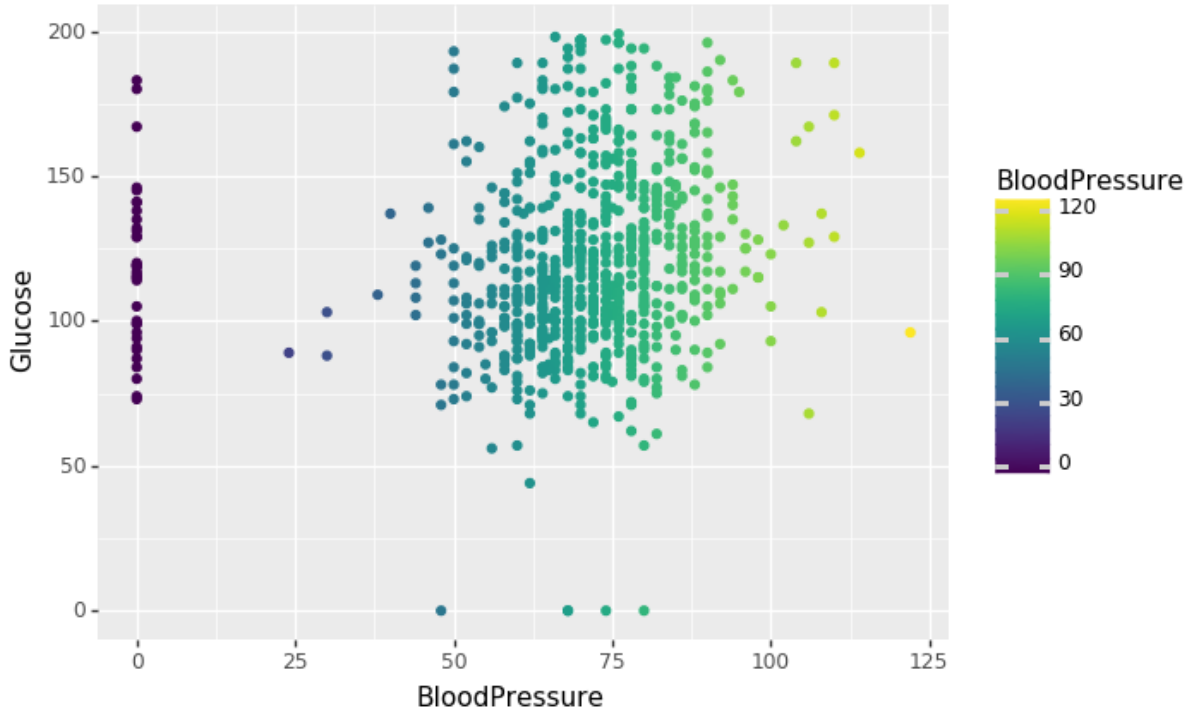
In [98]:
```
aye = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392ParlettPelleriti/master/Data/diabetes2.csv")

aye.head()
```

Out[98]:

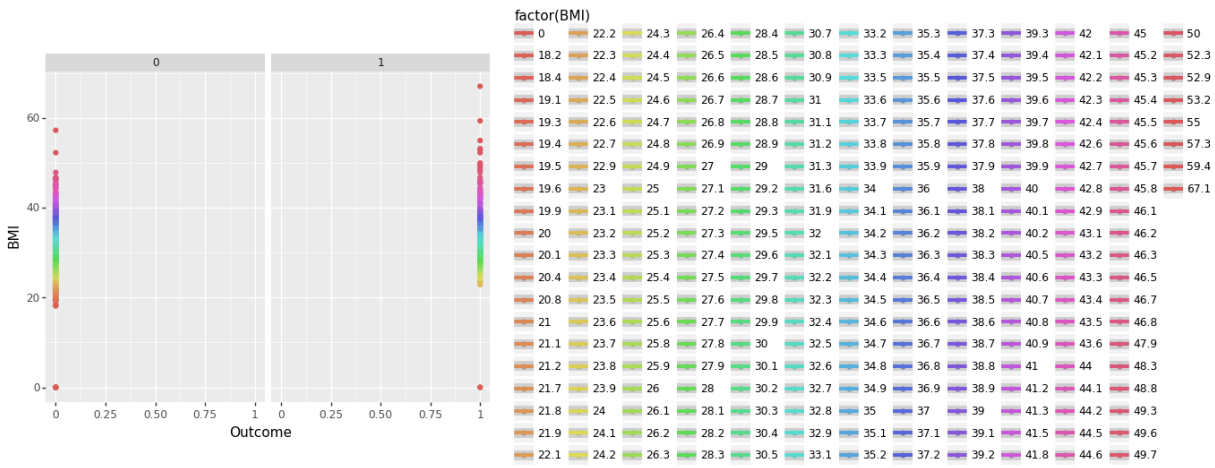
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [124]:
```
#1
(ggplot(aye, aes(x = "BloodPressure", y = "Glucose", color = "BloodPressure")) + geom_point())
```



Out[124]: <ggplot: (318286085)>

In [129]:
```
#2
(ggplot(aye, aes('Outcome', 'BMI', color = 'factor(BMI)'))
 + geom_point()
 + stat_smooth(method='lm')
 + facet_wrap('~Outcome'))
```



Out[129]: <ggplot: (312251937)>