Data Cleaning & analyzing

Data Visualizati on

Building a Machine Learning Model

Use CSV For collect data

Supervised Learning Project

Rinkal raj

Project/Goals

- Use supervised learning techniques to build a machine learning model that can predict whether a patient has diabetes or not, based on certain diagnostic measurements.
- The project involves three main parts: exploratory data analysis, preprocessing, feature engineering, and training a machine learning model.

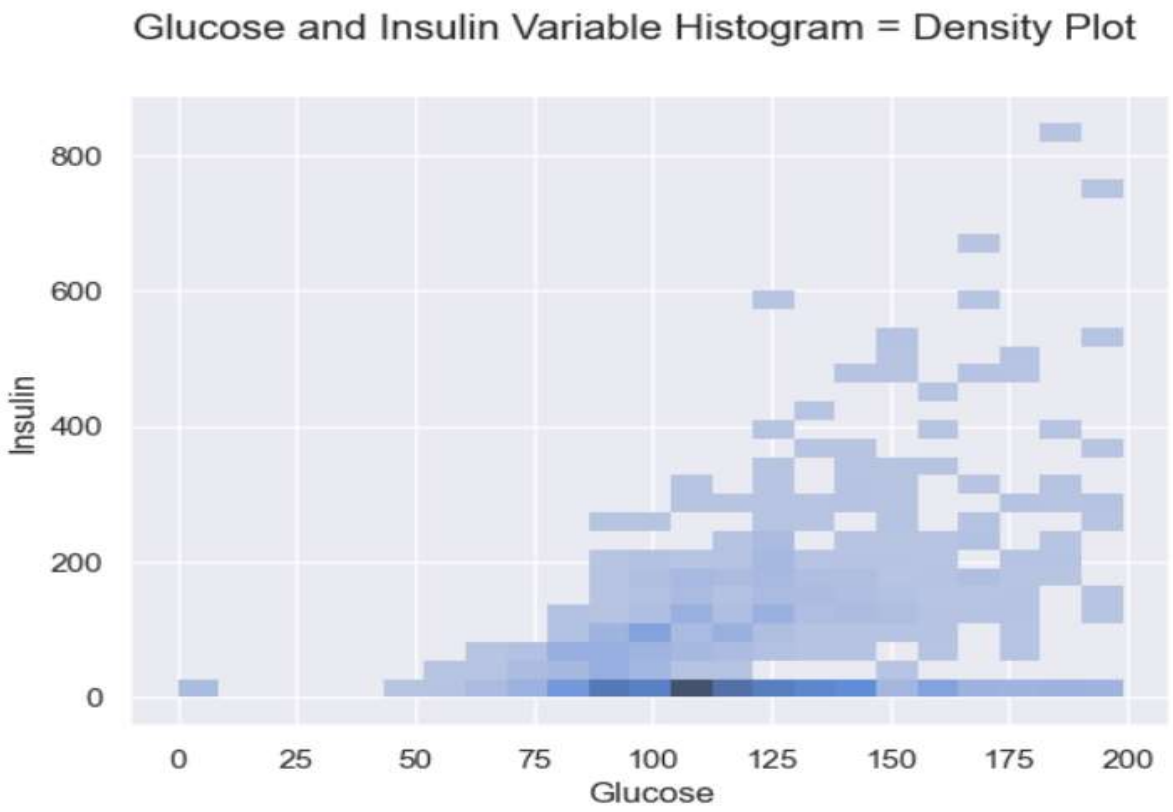
Tool Installation & Set Up

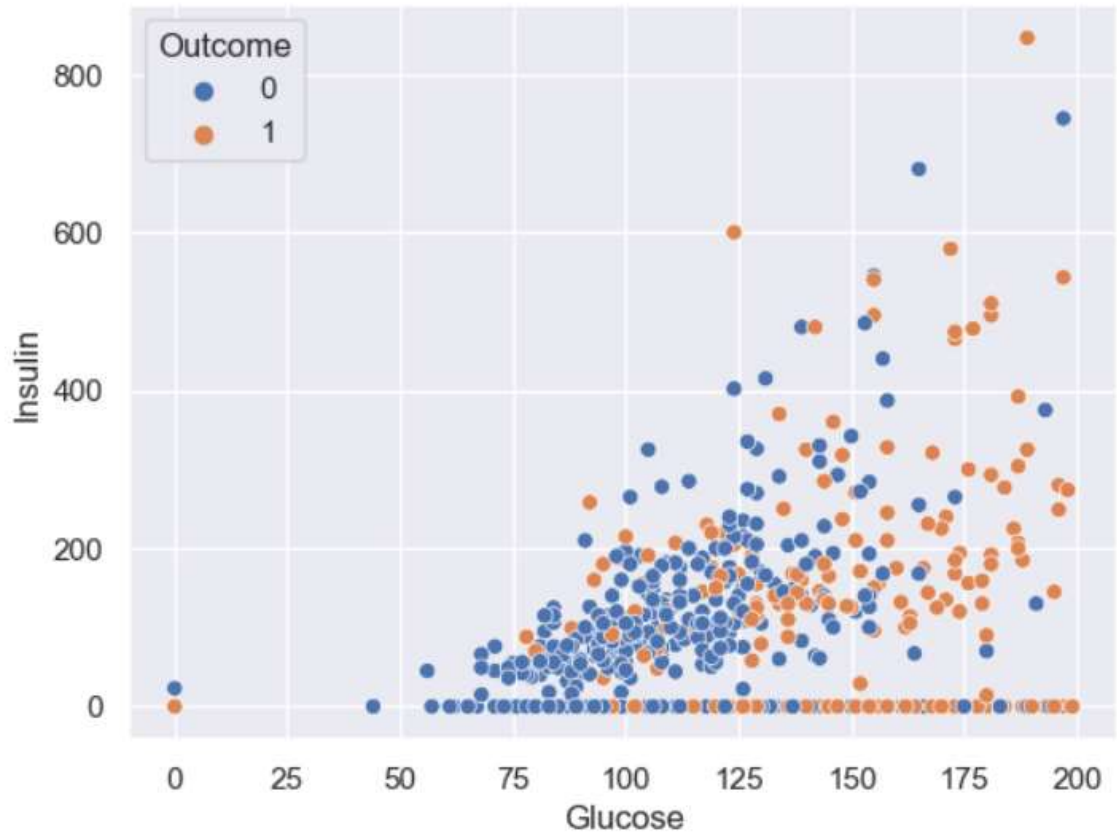
Use Jupyter lab for python programming.

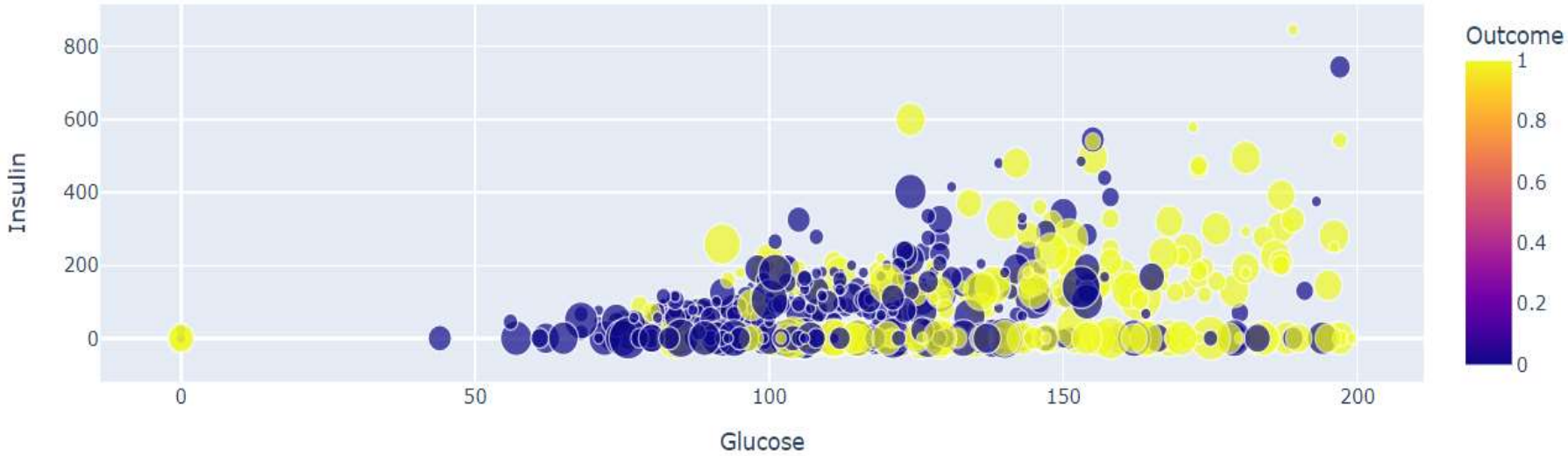
Use Visual Studio for editing files which clone from GitHub.

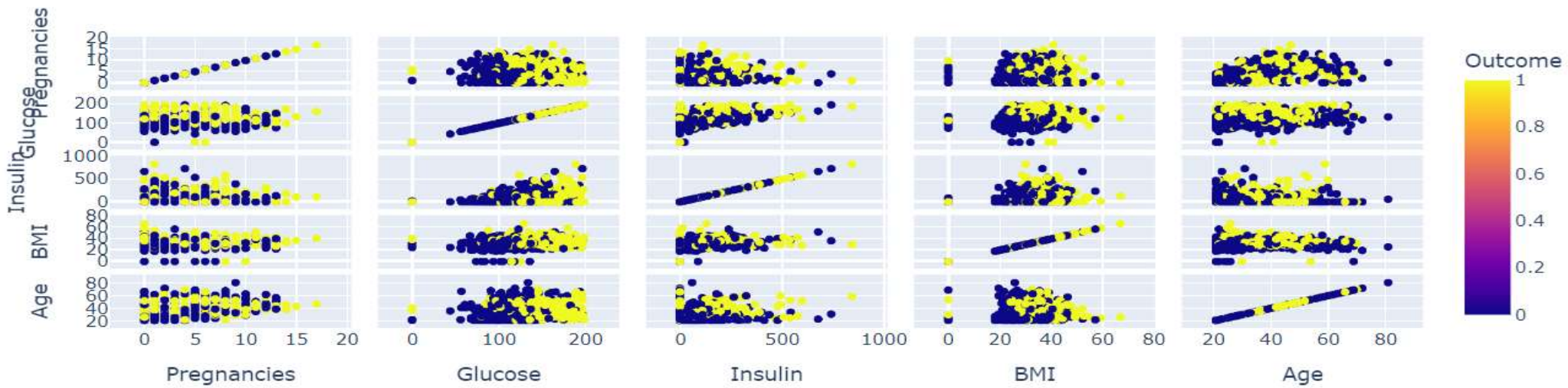
Use GitHub for storing files of project data and share on local server it is easy way to access and store data.

Relation between variables

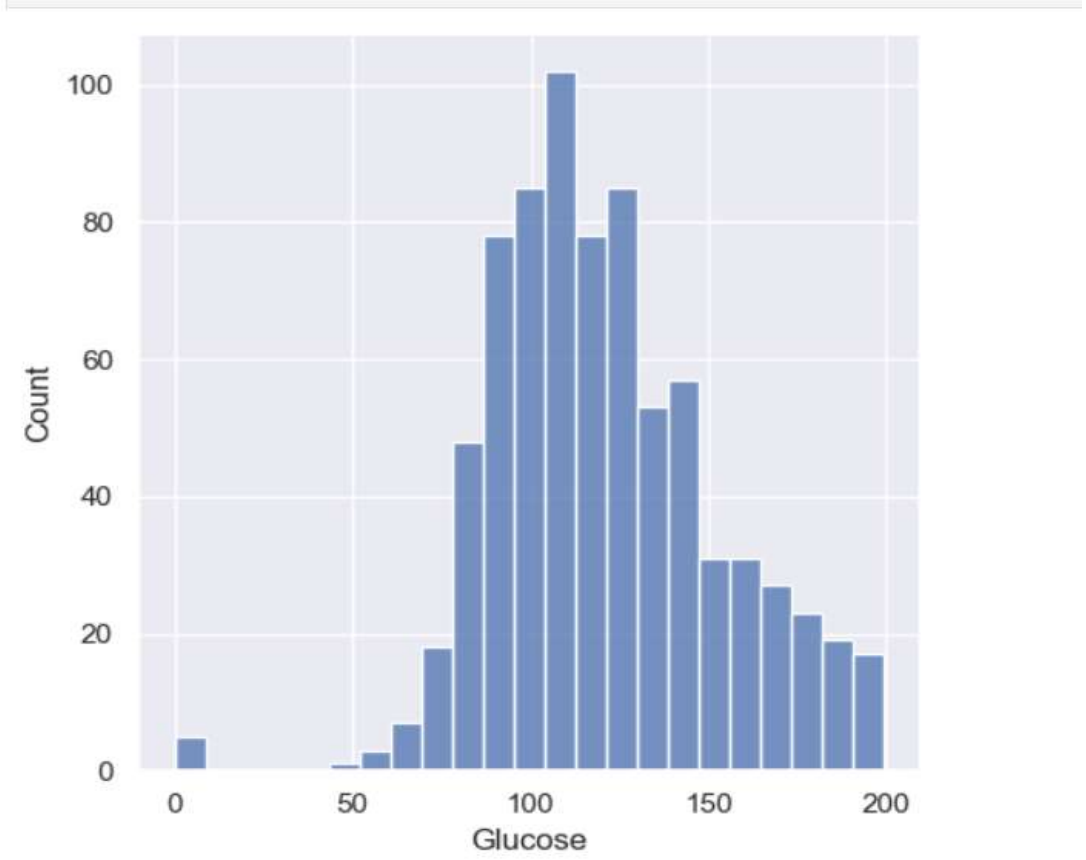


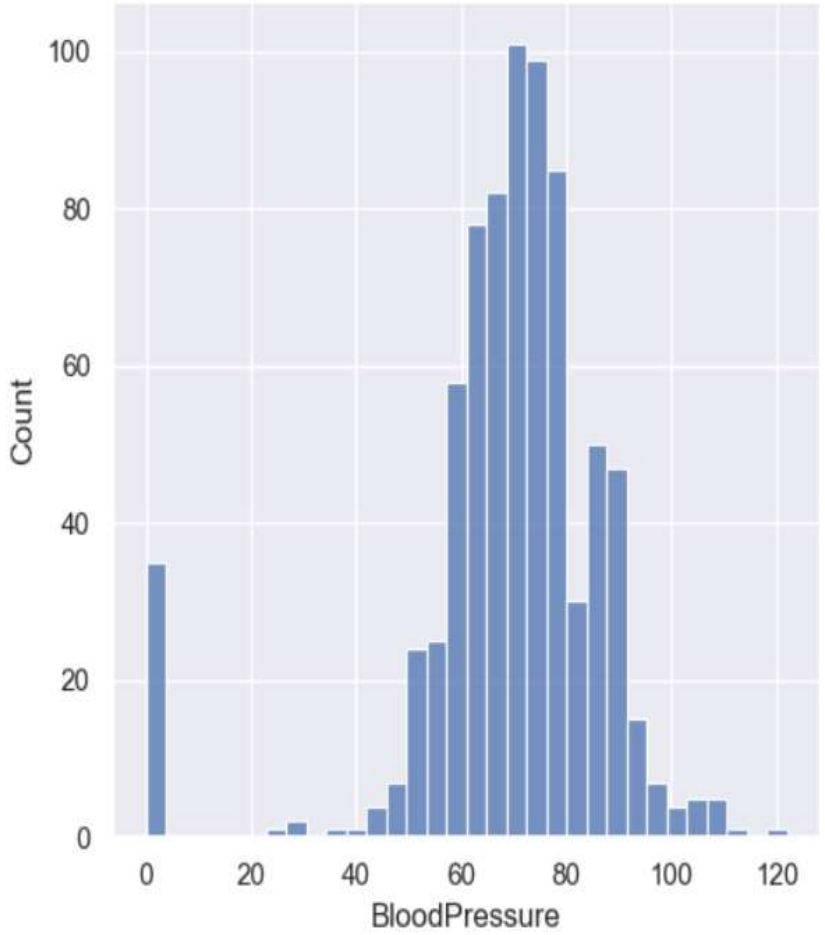

Outcome
0
1
800
600
400
200
0
Insulin
0
25
50
75
100
125
150
175
200
Glucose

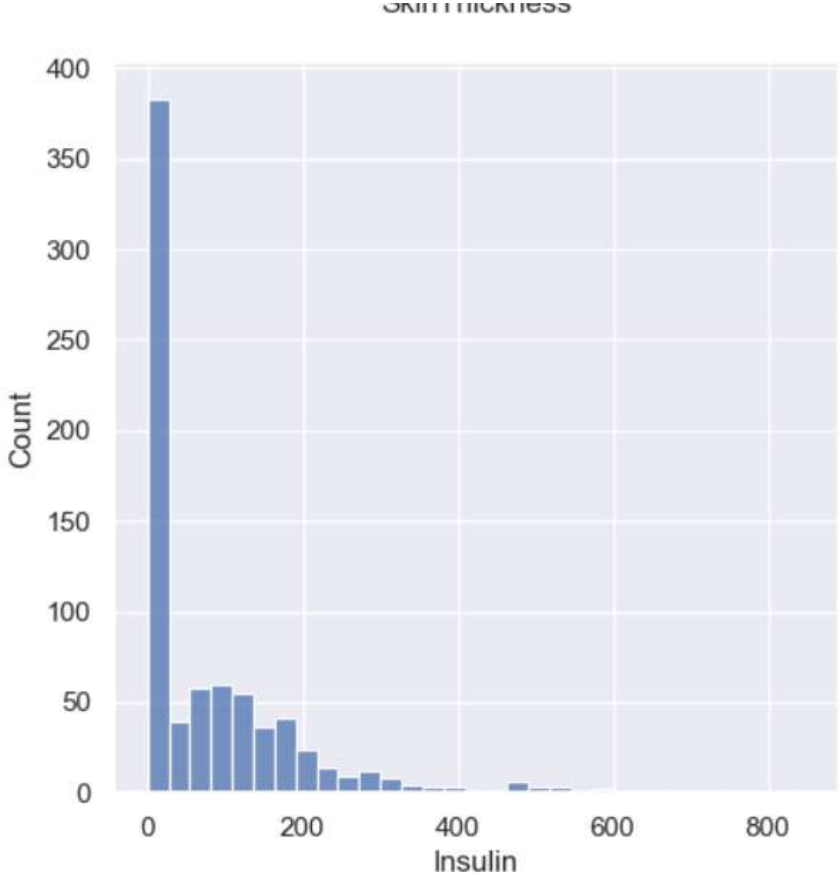

Insulin
800
600
400
200
0
0
50
100
150
200
Glucose
Outcome
1
0.8
0.6
0.4
0.2
0

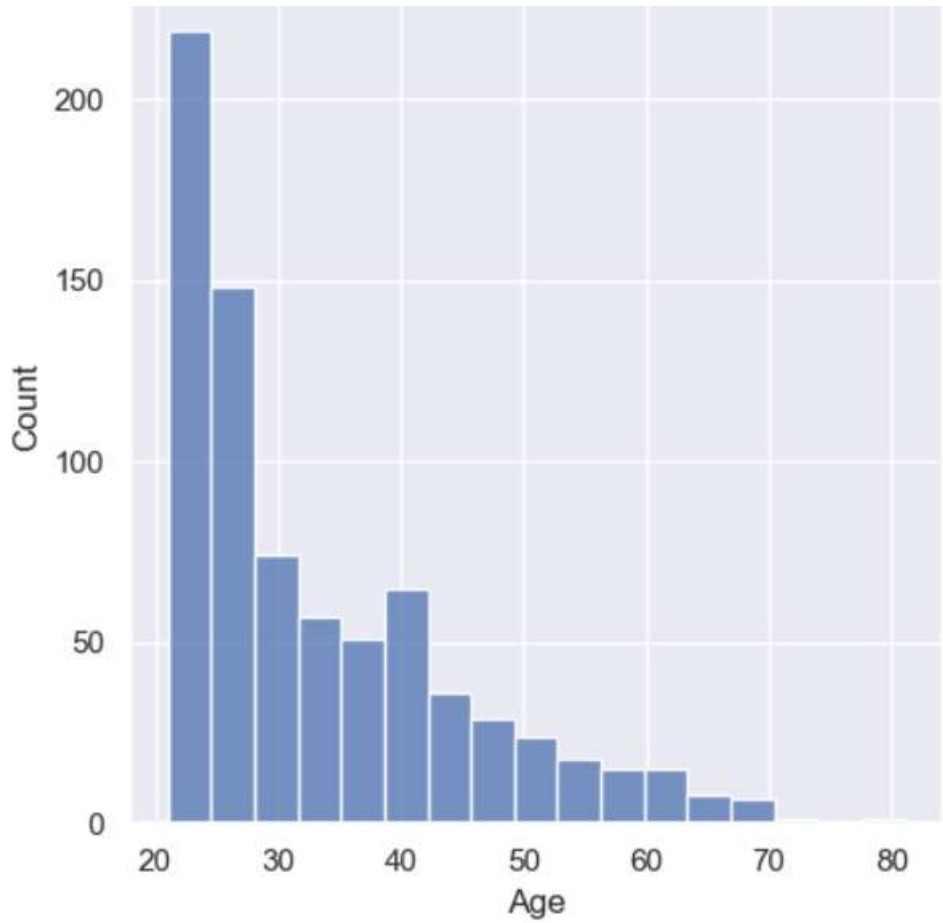

Pregnancies
Glucose
Insulin
BMI
Age
Pregnancies
Glucose
Insulin
BMI
Age
Outcome
1
0.8
0.6
0.4
0.2
0

Distribution of variable

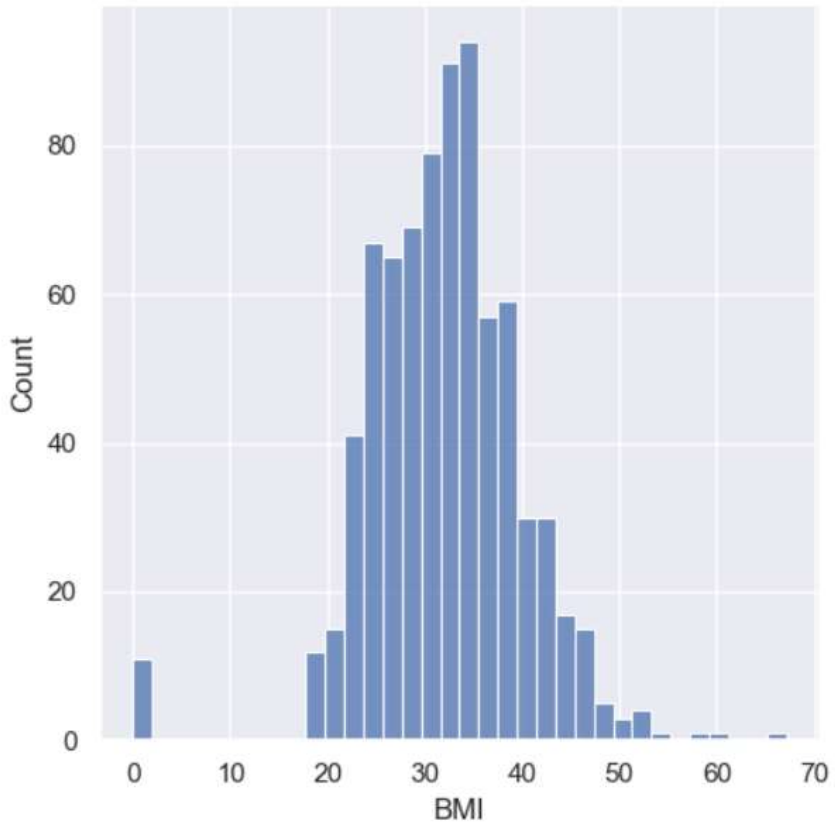


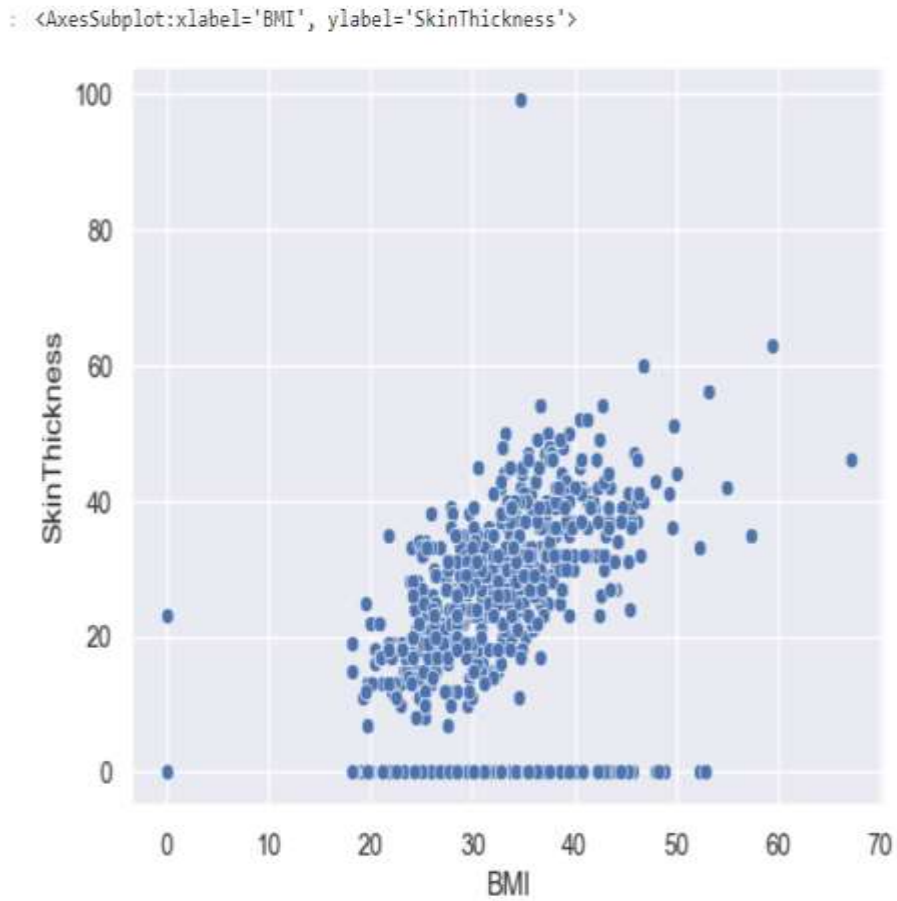


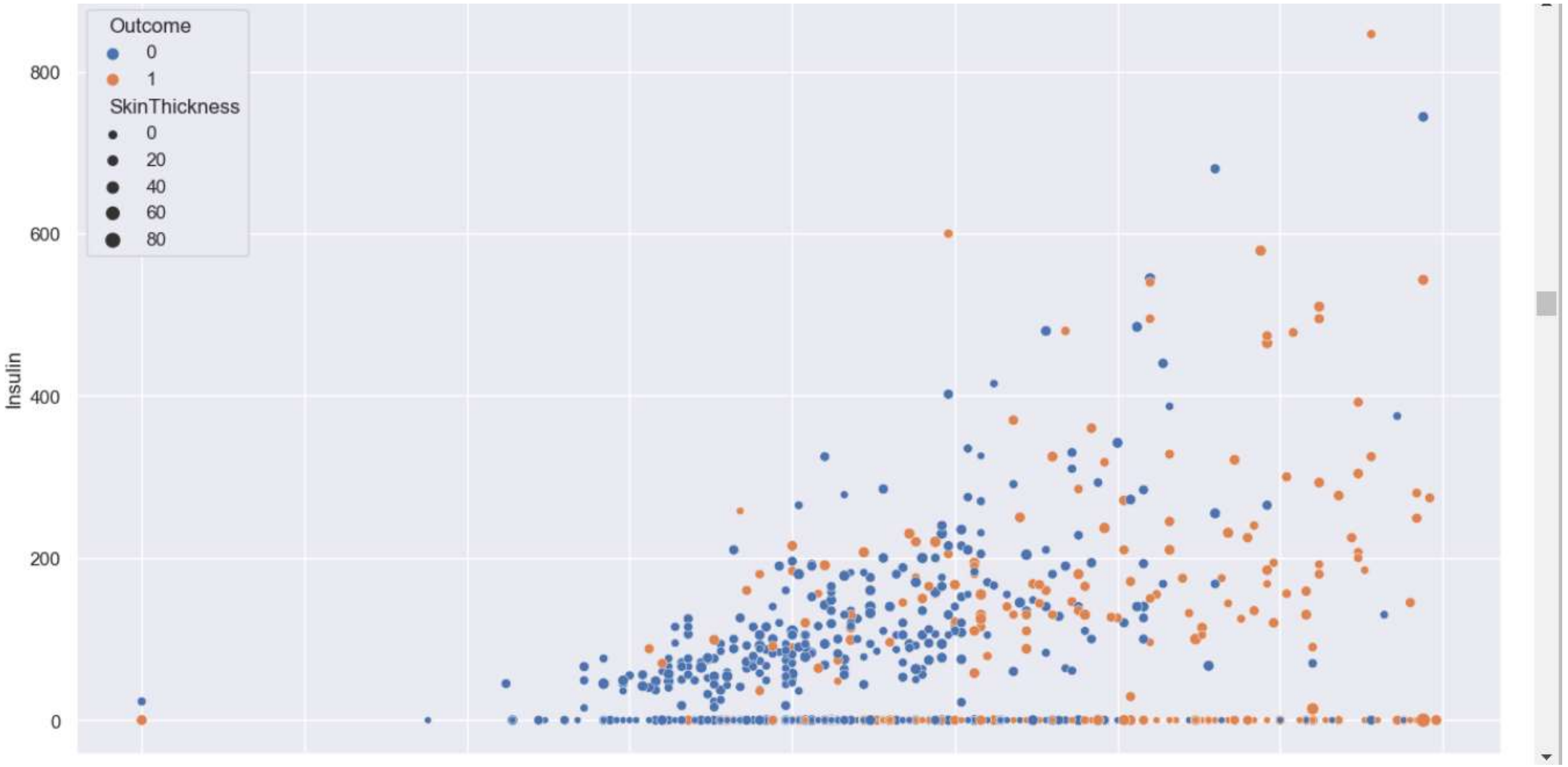

SkinThickness
400
350
300
250
200
150
100
50
0
Count
0
200
400
600
800
Insulin


200
150
100
50
0
Count
20
30
40
50
60
70
80
Age

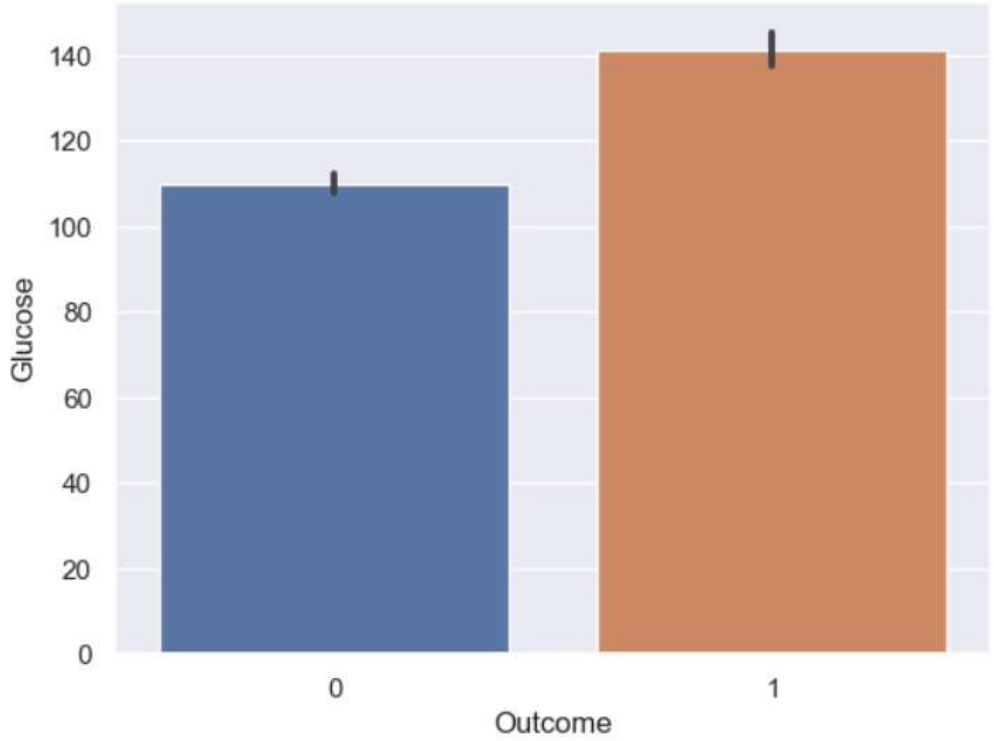
Outliners


Count
BMI
0
20
40
60
80
0
10
20
30
40
50
60
70

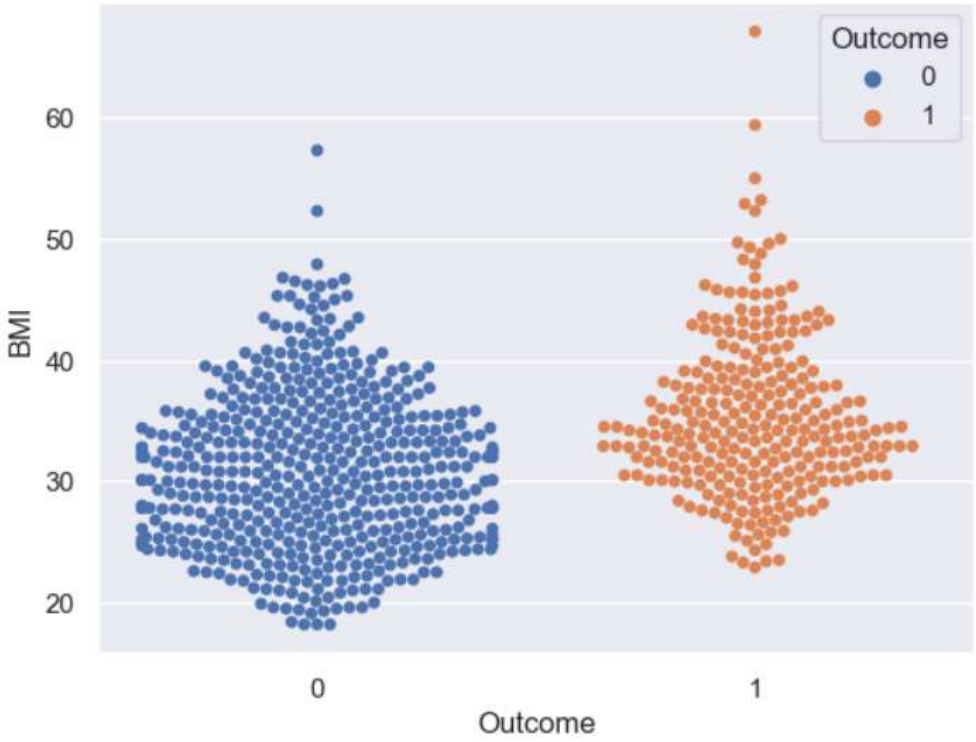

<AxesSubplot:xlabel='BMI', ylabel='SkinThickness'>
SkinThickness
BMI
0
20
40
60
80
100
0
10
20
30
40
50
60
70


Outcome
0
1
SkinThickness
0
20
40
60
80
Insulin
800
600
400
200
0

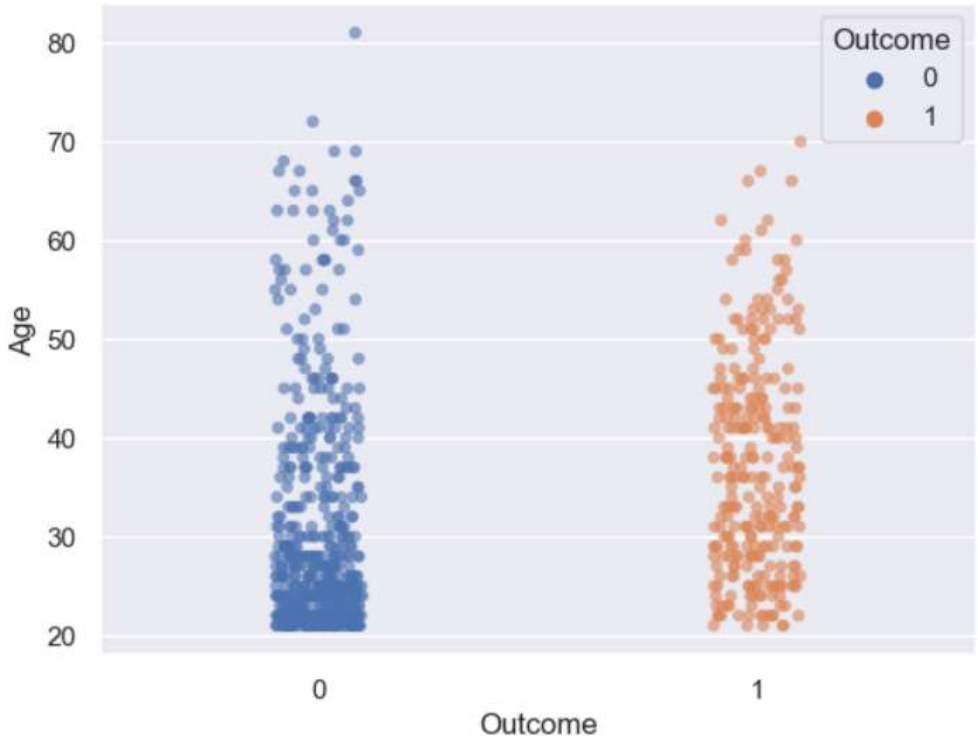
The average glucose level for individuals with diabetes and without diabetes



The average BMI for individuals with diabetes and without diabetes

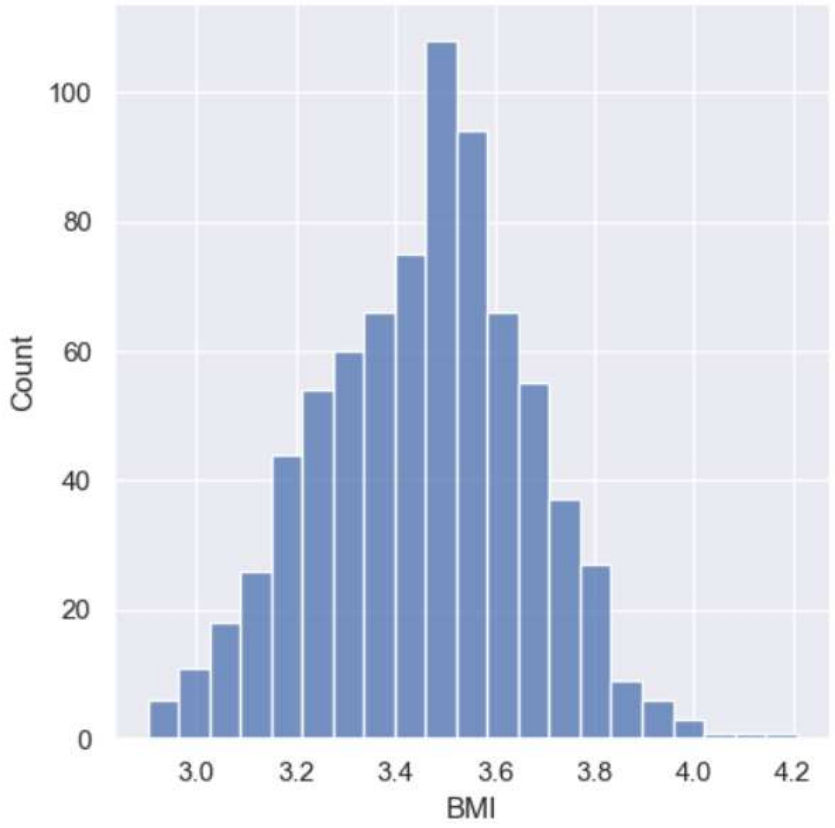


The average age

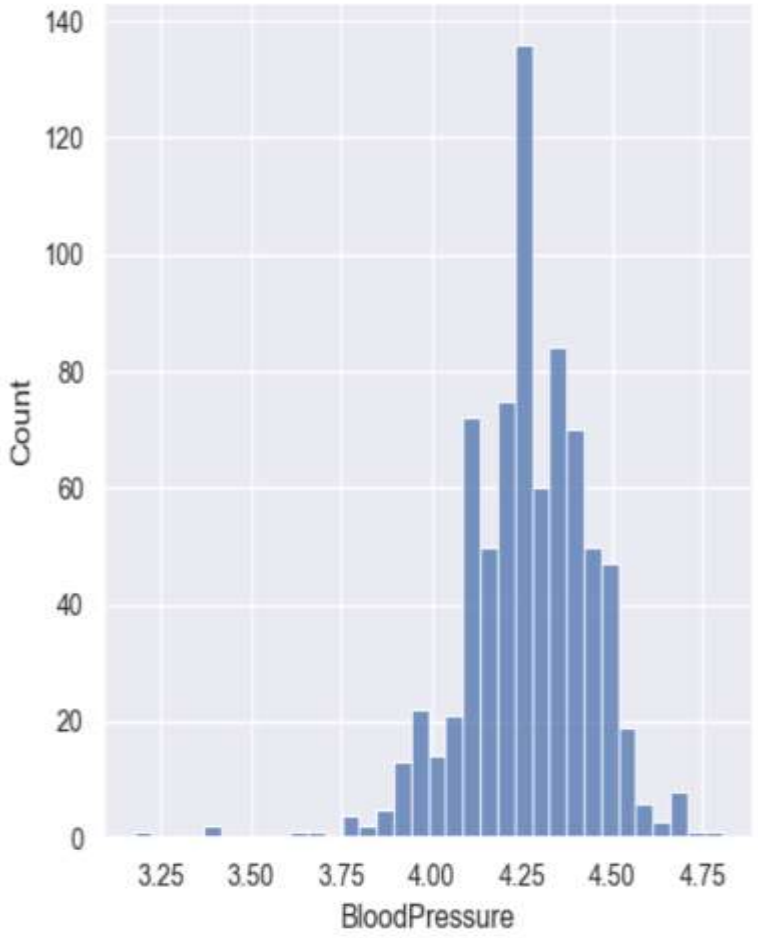


Log Transform

<seaborn.axisgrid.FacetGrid at 0x1b46057d508>



<seaborn.axisgrid.FacetGrid at 0x1b46517dac8>



0’s values replace by mean

```
#Replace 0 with Mean values
df['SkinThickness'] = df['SkinThickness'].replace(0,df['SkinThickness'].mean())
df['Pregnancies'] = df['Pregnancies'].replace(0,df['Pregnancies'].mean())
df['Glucose'] = df['Glucose'].replace(0,df['Glucose'].mean())
df['Insulin'] = df['Insulin'].replace(0,df['Insulin'].mean())
df['BloodPressure'] = df['BloodPressure'].replace(0,df['BloodPressure'].mean())
df['BMI'] = df['BMI'].replace(0,df['BMI'].mean())
```

Scaling and normalization

```
# Scaling
from sklearn.preprocessing import StandardScaler
#
scaler = StandardScaler()
df_train_scaled = pd.DataFrame(scaler.fit_transform(df[df_columns]))
```

```
df_train_scaled
```

	0	1	2	3	4	5
0	0.865276	-0.021044	0.872057	-0.417768	0.167255	1.425995
1	-1.205989	-0.516583	0.248678	-0.417768	-0.851535	-0.190672
2	2.015979	-0.681762	-0.630654	-0.417768	-1.331821	-0.105584
3	-1.074480	-0.516583	-0.374700	-0.265107	-0.633222	-1.041549
4	0.503626	-2.663916	0.872057	0.530423	1.549899	-0.020496
...	...	...	...	...	...	...
763	-0.679954	0.309315	2.222711	0.659428	0.065376	2.532136
764	0.010468	-0.186224	0.040885	-0.417768	0.632988	-0.531023
765	-0.022409	-0.021044	-0.374700	-0.071599	-0.909751	-0.275760
766	0.141977	-1.012121	-0.630654	-0.417768	-0.342140	1.170732
767	-0.942972	-0.186224	0.456471	-0.417768	-0.298477	-0.871374

[133]:
```
from sklearn.preprocessing import Normalizer

n = Normalizer()
n.fit(df_copy)
df_norm = pd.DataFrame(n.transform(df_copy),columns = df_copy.columns)
```

[134]:
```
sns.pairplot(df_norm)
```

Logistic Model

```
|: from sklearn.linear_model import LogisticRegression
   from sklearn.model_selection import train_test_split


   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

   log_reg = LogisticRegression()
   log_reg.fit(X_train, y_train)

   acc_train = log_reg.score(X_train,y_train)
   print(f'acc train: {acc_train}')

   acc_test = log_reg.score(X_test,y_test)
   print(f'acc test: {acc_test}')
```

```
acc train: 0.7638436482084691
acc test: 0.7857142857142857
```

The model is good fit model because the ratio of testing and training data is almost similar.

```
df['predicted_outcome'] = (df.Outcome >= 0).astype('int')
# ground truth
y_true = df.Outcome.values

# simulate probabilites of positive class
y_proba =  df.predicted_outcome.values

# set the threshold to predict positive class
thres = 0.5

# class predictions
y_pred = [int(value > thres) for value in y_proba]
```

```
# import confusion_matrix from sklearn
from sklearn.metrics import confusion_matrix

# compute confusion_matrix
confusion_matrix = confusion_matrix(y_true,y_pred)

print(confusion_matrix)
```

```
[[  0 500]
 [  0 268]]
```

```
# import accuracy_score from sklearn
from sklearn.metrics import accuracy_score

# compute accuracy
accuracy = accuracy_score(y_true,y_pred)

# print accuracy
#print(accuracy)
print('Accuracy Score: %.3f'%(accuracy))
```

```
Accuracy Score: 0.349
```

```
# import f1_score from sklearn
from sklearn.metrics import f1_score

# compute F1-score
f1_score = f1_score(y_true,y_pred)

# print F1-score
#print(f1_score)
print('F1-Score: %.3f'%(f1_score))
```

```
F1-Score: 0.517
```

Ensemble Model

```
[159]: # import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

# instantiate the classifier

rfc = RandomForestClassifier(random_state=0)



# fit the model

rfc.fit(X_train, y_train)
# Predict the Test set results

y_pred = rfc.predict(X_test)



# Check accuracy score

from sklearn.metrics import accuracy_score

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score with 10 decision-trees : 0.7403
```

[161]:
```
# Print the Confusion Matrix and slice it into four pieces

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)
```

```
Confusion matrix

 [[78 18]
 [22 36]]
```

[162]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.81      0.80        96
           1       0.67      0.62      0.64        58

    accuracy                           0.74       154
   macro avg       0.72      0.72      0.72       154
weighted avg       0.74      0.74      0.74       154
```

Conclusion

- In this project, I built various EDA for data exploration and understanding of datasets. Thus, I found Glucose, BMI, skin Thickness, Blood Presser and age are the most predictable variable.
- I found a person who has a high glucose level and BMI compares to the average level of a person who has diabetes.
- I also created Logistic Regression and Ensemble Random Forest Classifier machine learning model and compares these models I found Ensemble Random Forest Classifier is the best-fit model because the accuracy score is too low for a logistic regression model.
- The model accuracy score with 10 decision trees is 0.743.
- Confusion matrix and classification report are other tools to visualize the model performance. They yield good performance.

THANK YOU