Healthcare PGP – Project 2

DESCRIPTION

NIDDK (National Institute of Diabetes and Digestive and Kidney Diseases) research creates knowledge about and treatments for the most chronic, costly, and consequential diseases.

- The dataset used in this project is originally from NIDDK. The objective is to predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.
- Build a model to accurately predict whether the patients in the dataset have diabetes or not.

Dataset Description

The datasets consists of several medical predictor variables and one target variable (Outcome). Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and more.

Variables	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration in an oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skinfold thickness (mm)
Insulin	Two hour serum insulin
BMI	Body Mass Index
DiabetesPedigreeFunction	Diabetes pedigree function
Age	Age in years
Outcome	Class variable (either 0 or 1). 268 of 768 values are 1, and the others are 0

Project Task: Week 1

Data Exploration:

1. Perform descriptive analysis. Understand the variables and their corresponding values. On the columns below, a value of zero does not make sense and thus indicates missing value:

- Glucose
- BloodPressure
- SkinThickness
- Insulin
- BMI

2. Visually explore these variables using histograms. Treat the missing values accordingly.

3. There are integer and float data type variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

Data Exploration:

4. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.

5. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

6. Perform correlation analysis. Visually explore it using a heat map.

Project Task: Week 2

Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.
2. Apply an appropriate classification algorithm to build a model.
3. Compare various models with the results from KNN algorithm.
4. 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.

Python:

Importing / Data Wrangling / EDA / Machine Learning

Healthcare PGP

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt import
seaborn as sns
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('health care diabetes.csv') df
```

	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]

Descriptive Analysis

df.isna().sum()

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

int64

df.info()

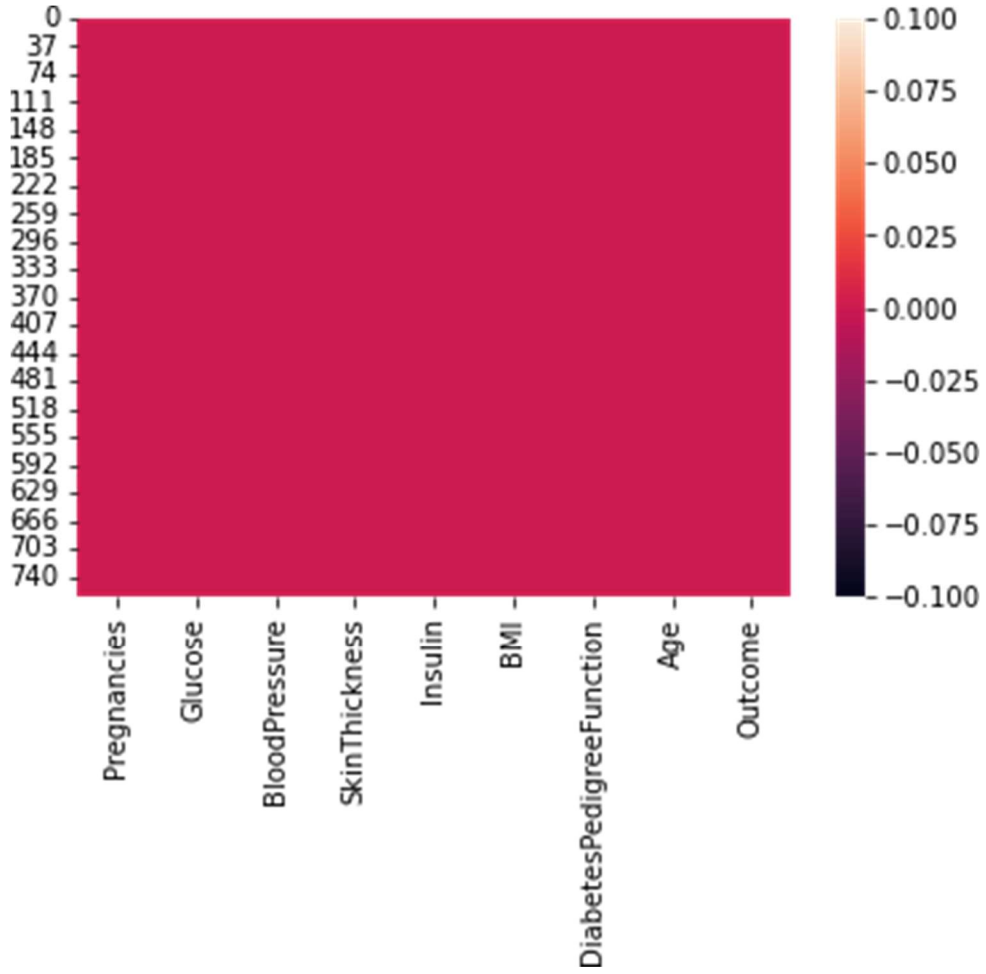
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767 Data columns
(total 9 columns):

#	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

sns.heatmap(df.isna())

<AxesSubplot:>



df.describe().T

	count	mean	std	min	25% \
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000
Glucose	768.0	120.894531	31.972618	0.000	99.00000
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000
Insulin	768.0	79.799479	115.244002	0.000	0.00000
BMI	768.0	31.992578	7.884160	0.000	27.30000
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375
Age	768.0	33.240885	11.760232	21.000	24.00000

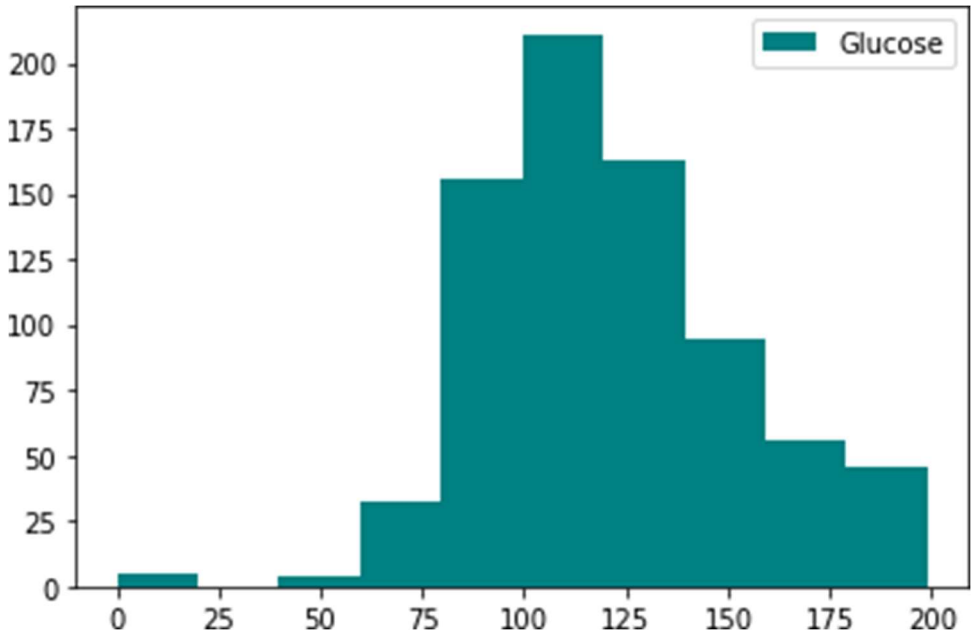
Outcome	768.0	0.348958	0.476951	0.000
0.00000				

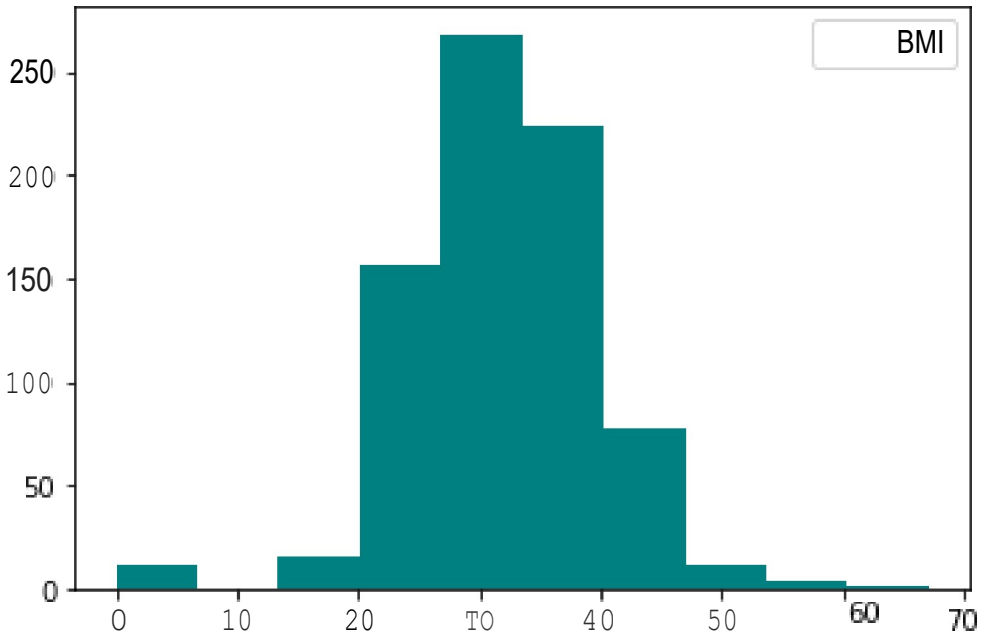
	50%	75%	max
Pregnancies	3.0000	6.00000	17.00
Glucose	117.0000	140.25000	199.00
BloodPressure	72.0000	80.00000	122.00
SkinThickness	23.0000	32.00000	99.00
Insulin	30.5000	127.25000	846.00
BMI	32.0000	36.60000	67.10
DiabetesPedigreeFunction	0.3725	0.62625	2.42
Age	29.0000	41.00000	81.00
Outcome	0.0000	1.00000	1.00

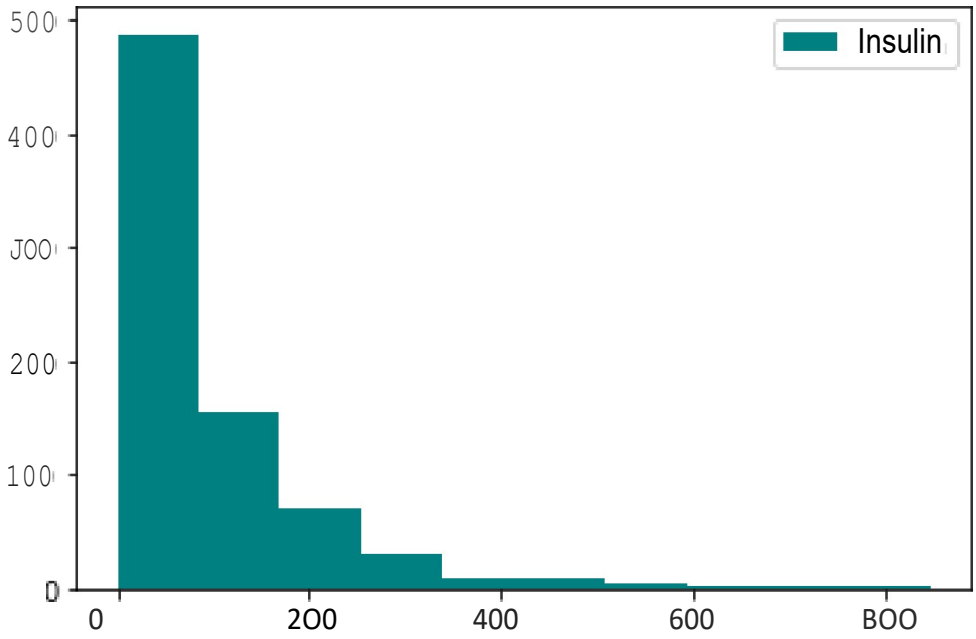
Visual Exploration of Data

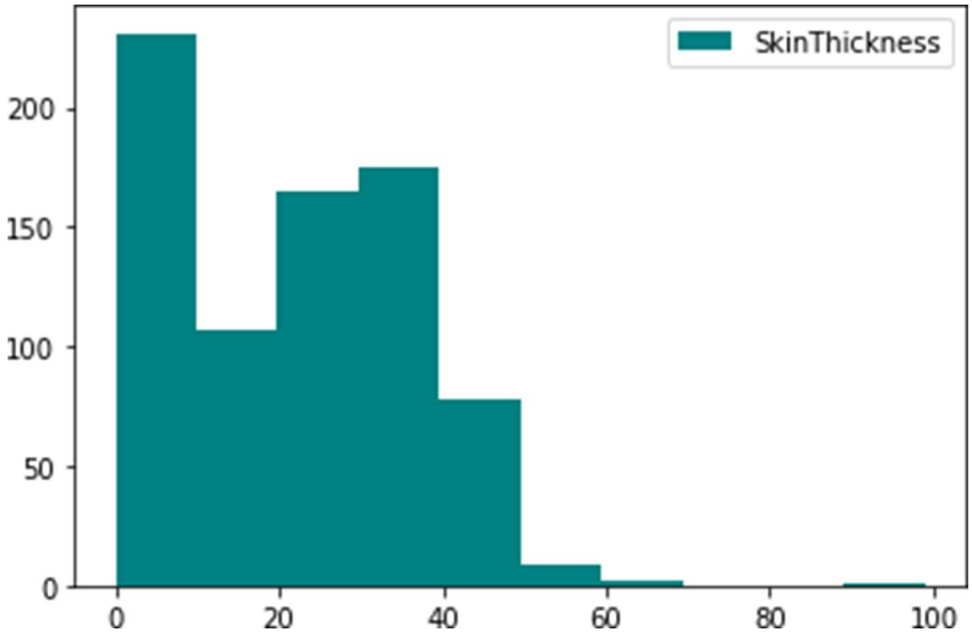
Distribution of data and their skewness.

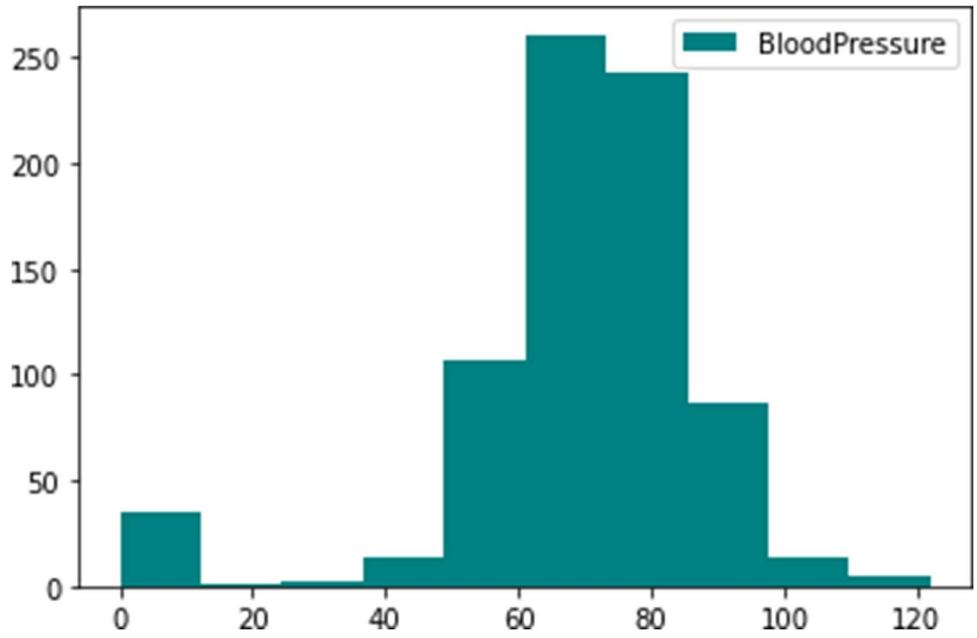
```
for i in ['Glucose', 'BMI', 'Insulin', 'SkinThickness', 'BloodPressure']:

    plt.hist(df[i], color = 'teal', label = i) plt.legend()
    plt.show()
```




BMI
250
200
150
100
50
0
0
10
20
TO
40
50
60
70


Insulin
500
400
JOO
200
100
0
0
2O0
400
600
BOO





Indexes with Erroneous Values

```
for i in ['Glucose', 'BMI', 'Insulin', 'SkinThickness', 'BloodPressure']:
    print(df.loc[df[i] == 0].index)
```

```
Int64Index([75, 182, 342, 349, 502], dtype='int64')
Int64Index([9, 49, 60, 81, 145, 371, 426, 494, 522, 684, 706],
dtype='int64')
Int64Index([  0,   1,   2,   5,   7,   9,  10,  11,  12,  15,
```

```
                ...
                754, 756, 757, 758, 759, 761,       762,  764,  766,  767],
               dtype='int64', length=374)
Int64Index([      2,      5,      7,      9,     10,     11,     12,     15,     17,     21,
                ...
                734, 739, 743, 749, 750, 757,       758,  759,  762,  766],
               dtype='int64', length=227)
Int64Index([      7,     15,     49,     60,     78,     81,    172,    193,    222,    261,    266,
269, 300,
                332, 336, 347, 357, 426, 430,       435,  453,  468,  484,  494,
522, 533,
                535, 589, 601, 604, 619, 643,       697,  703,  706],
               dtype='int64')
```

Treating erroneous data with Median

```
for i in ['Glucose', 'BMI', 'Insulin', 'SkinThickness', 'BloodPressure']:
    df[i] = df[i].replace(0, df[i].median()) df.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    float64
 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(3), int64(6)
memory usage: 54.1 KB
```

```
df.iloc[75]
```

```
Pregnancies                 1.00
Glucose                   117.00
BloodPressure              48.00
SkinThickness              20.00
Insulin                    30.50
BMI                        24.70
DiabetesPedigreeFunction    0.14
Age                        22.00
Outcome                     0.00
Name: 75, dtype: float64
```

Checking if the values are applied

```
df[:1]
```

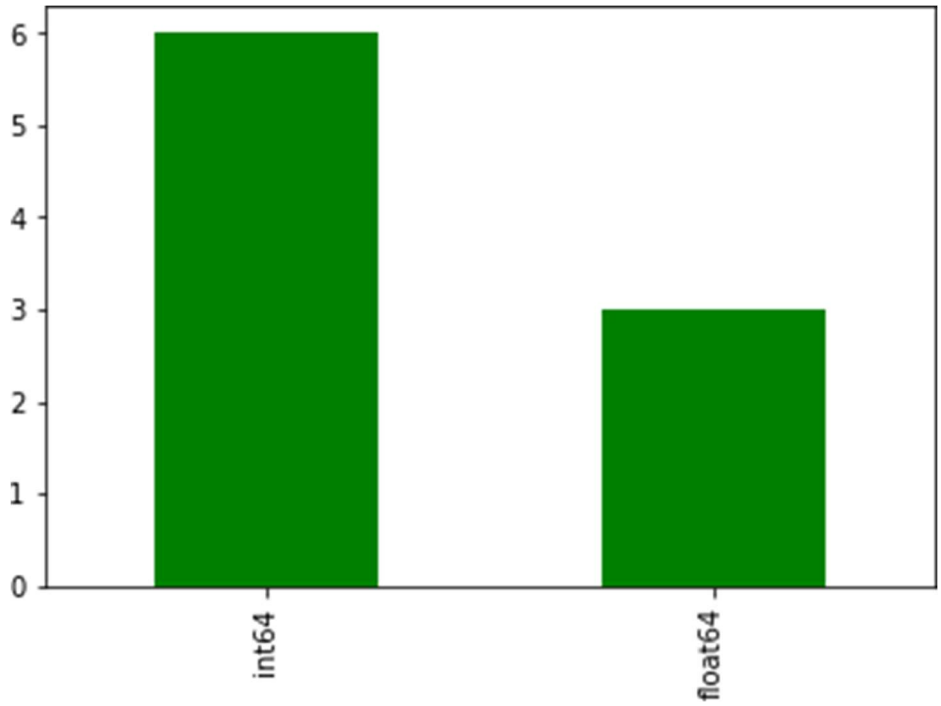
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  \
BMI
0            6      148             72             35        30.5   33.6

   DiabetesPedigreeFunction  Age  Outcome 0
                             0.627   50      1
```

```
df.nunique()
```

```
Pregnancies                 17
Glucose                    135
BloodPressure               46
SkinThickness               50
Insulin                    186
BMI                        247
DiabetesPedigreeFunction   517
Age                         52
Outcome                      2
dtype: int64
```

Data Types and the Count of Variables

```
df.dtypes.value_counts().plot(kind = 'bar', color = 'green').grid(False)
```



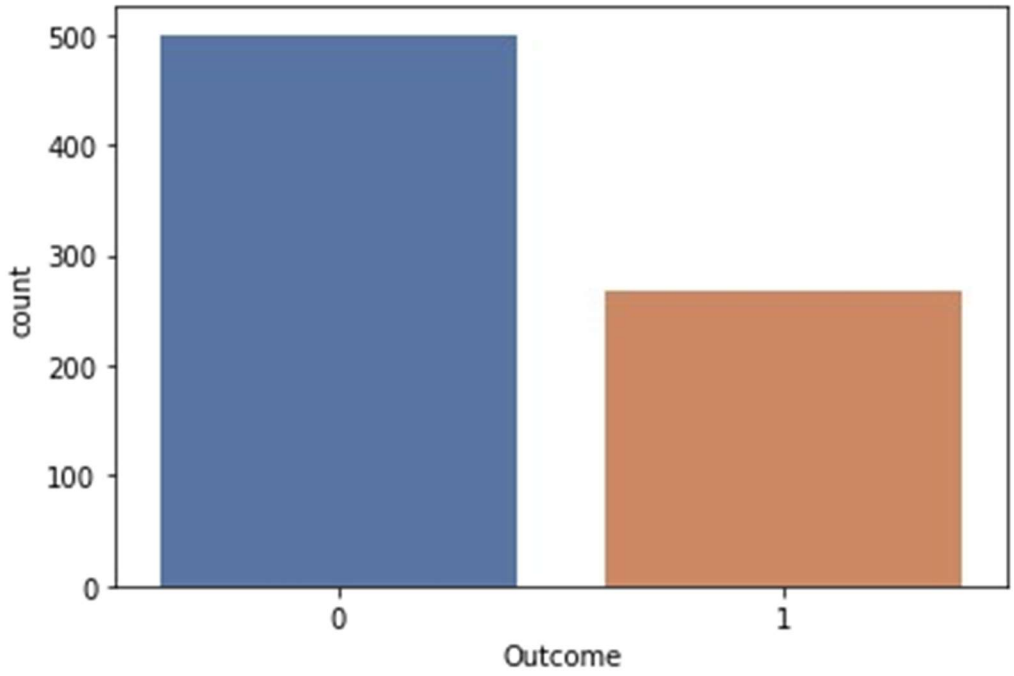
```
# df.to_csv('health care diabetes(No error).csv', index = False)
```

EDA

```
df['Outcome'].value_counts(normalize = True)
```

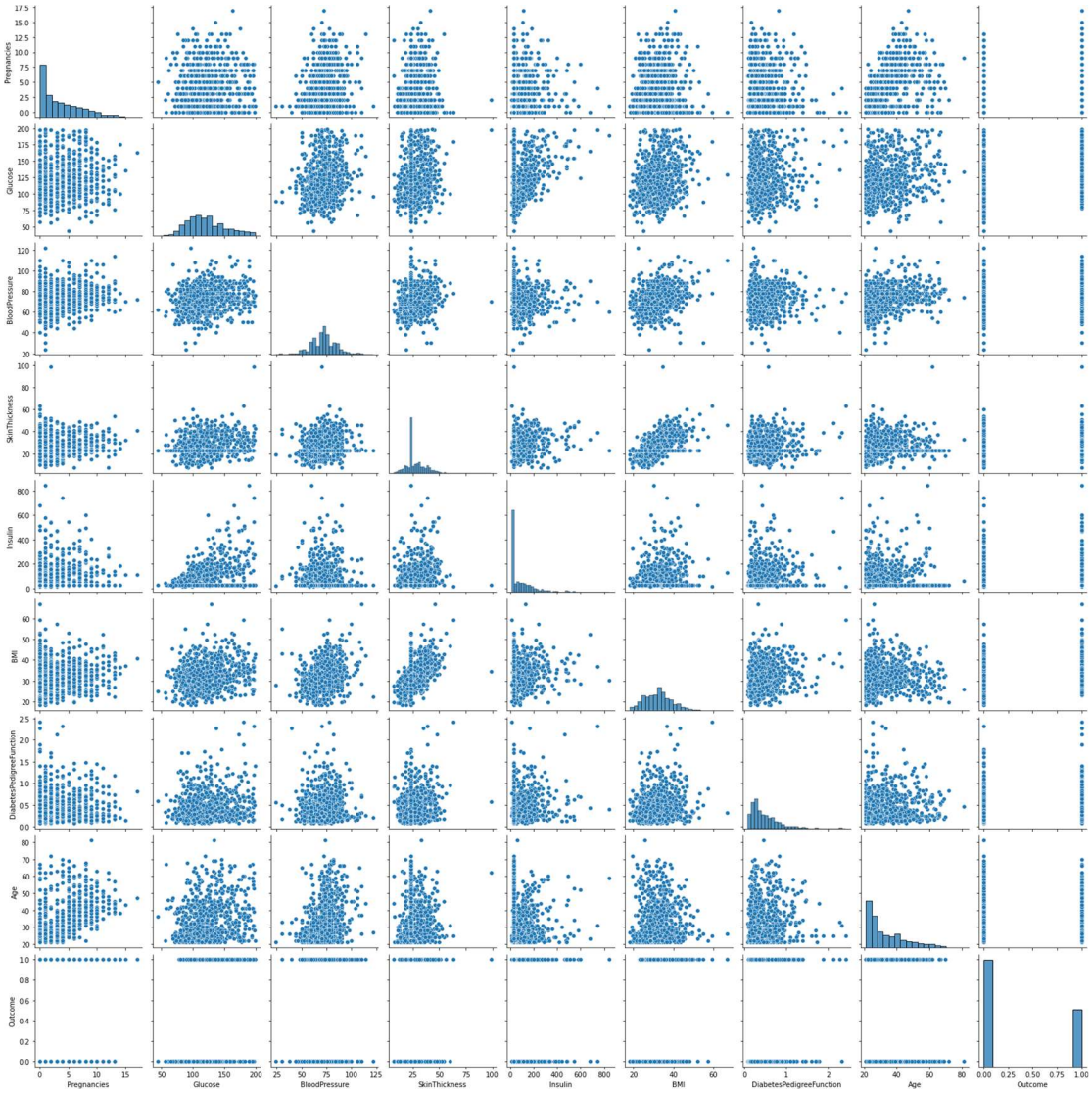
```
0       0.651042
1       0.348958
Name: Outcome, dtype: float64 sns.countplot(df['Outcome'], palette =

'deep')
```

<AxesSubplot:xlabel='Outcome', ylabel='count'>




```
sns.pairplot(df)
```

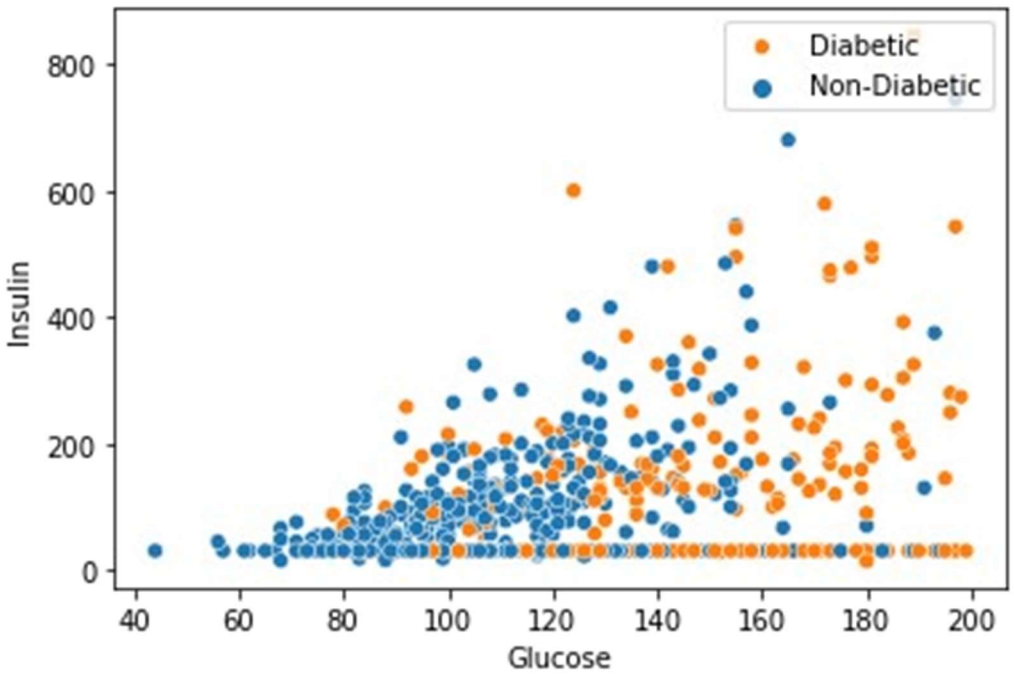
<seaborn.axisgrid.PairGrid at 0x20ba3c55580>



Individuals with Glucose levels greater than 140 are more susceptible of Diabetic conditions.

```
sns.scatterplot(data = df, x = 'Glucose', y = 'Insulin', hue = 'Outcome')
plt.legend(loc = 'upper right', labels = ['Diabetic','Non-Diabetic'])
```

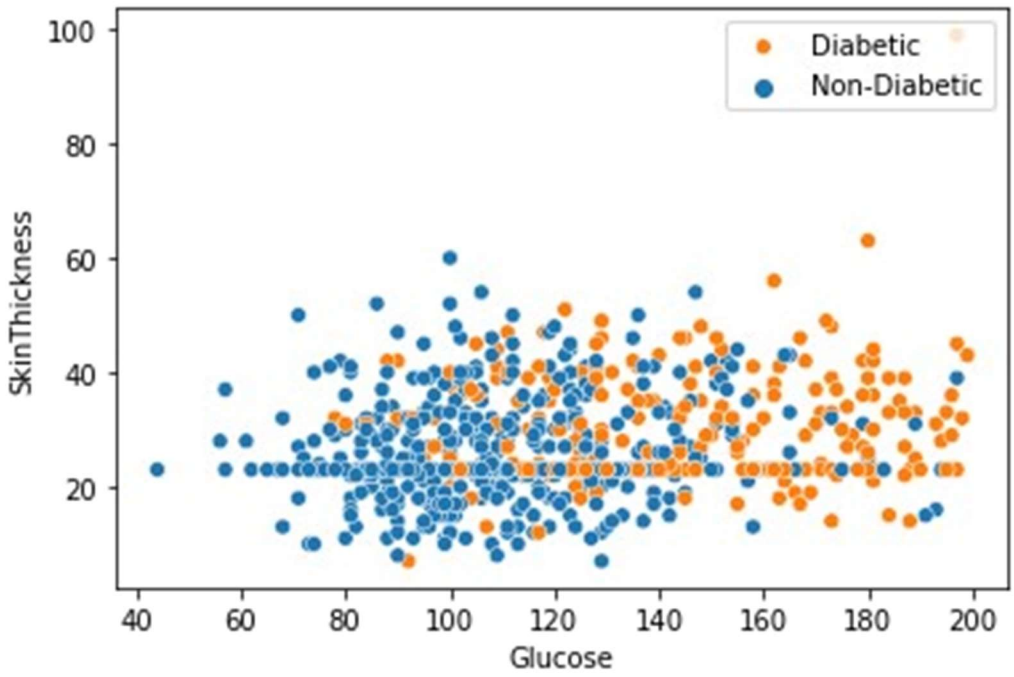
```
<matplotlib.legend.Legend at 0x20ba840f5b0>
```



Individulas with greater SkinThickness show higher diabetic conditions.

```
sns.scatterplot(data = df, x = 'Glucose', y = 'SkinThickness', hue = 'Outcome')
plt.legend(loc = 'upper right', labels = ['Diabetic','Non-Diabetic'])
```

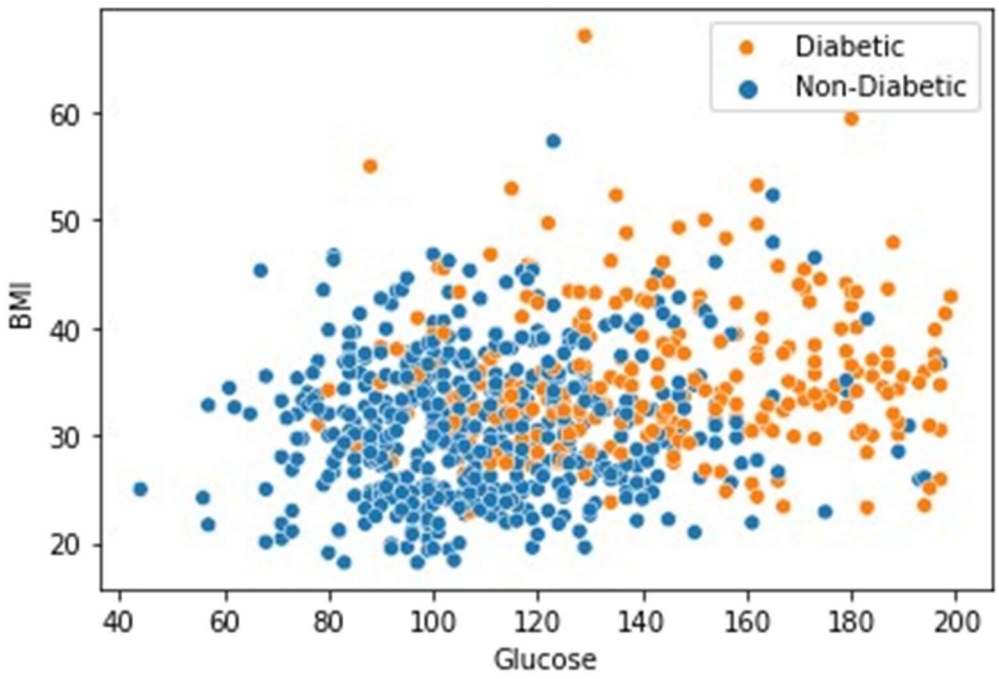
```
<matplotlib.legend.Legend at 0x20ba8eda1c0>
```



A higher BMI could be very well be a case of Diabetes.

```
sns.scatterplot(data = df, x = 'Glucose', y = 'BMI', hue = 'Outcome') plt.legend(loc = 'upper right', labels = ['Diabetic','Non-Diabetic'])
```

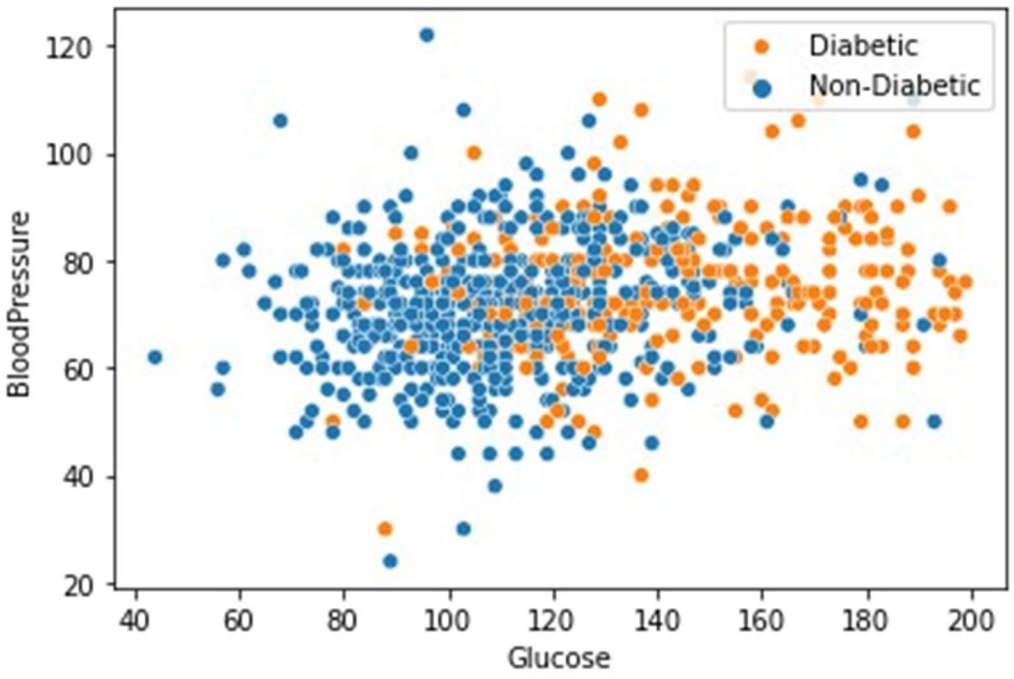
<matplotlib.legend.Legend at 0x20ba8f2db80>



Diabetic patients do show a greater blood pressure levels.

```
sns.scatterplot(data = df, x = 'Glucose', y = 'BloodPressure', hue = 'Outcome')
plt.legend(loc = 'upper right', labels = ['Diabetic','Non-Diabetic'])
```

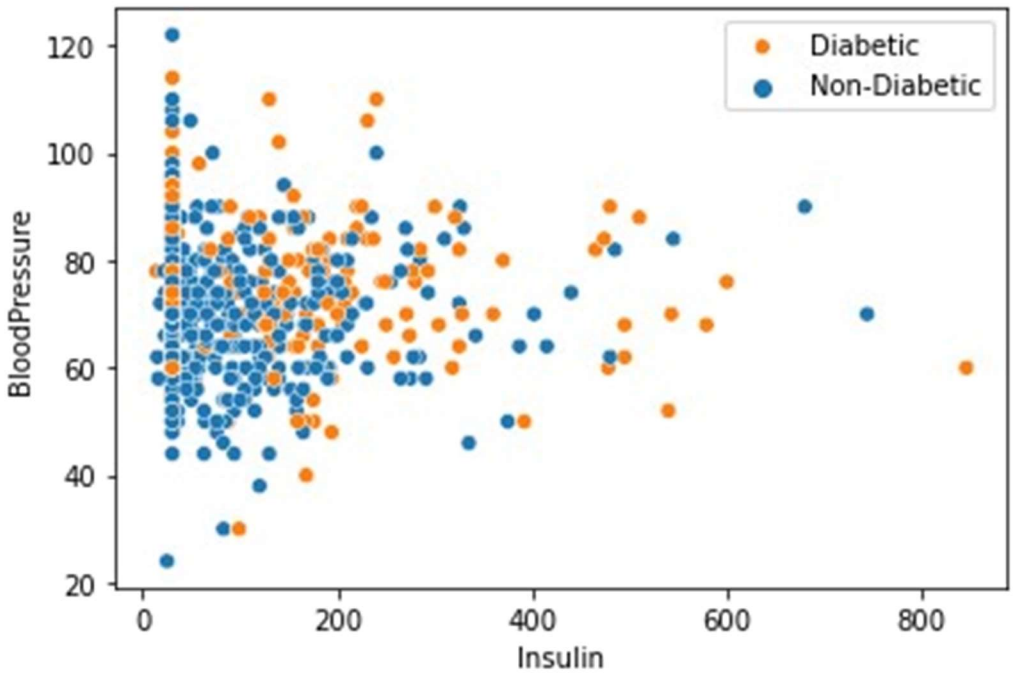
<matplotlib.legend.Legend at 0x20ba8fa7310>



Higher Blood Pressure doesn't necessarily mean a spike in Insulin Levels.

```
sns.scatterplot(data = df, x = 'Insulin', y = 'BloodPressure', hue = 'Outcome')
plt.legend(loc = 'upper right', labels = ['Diabetic','Non-Diabetic'])
```

```
<matplotlib.legend.Legend at 0x20ba84de0a0>
```



df.corr() SkinThickness

	Pregnancies	Glucose	BloodPressure	\
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

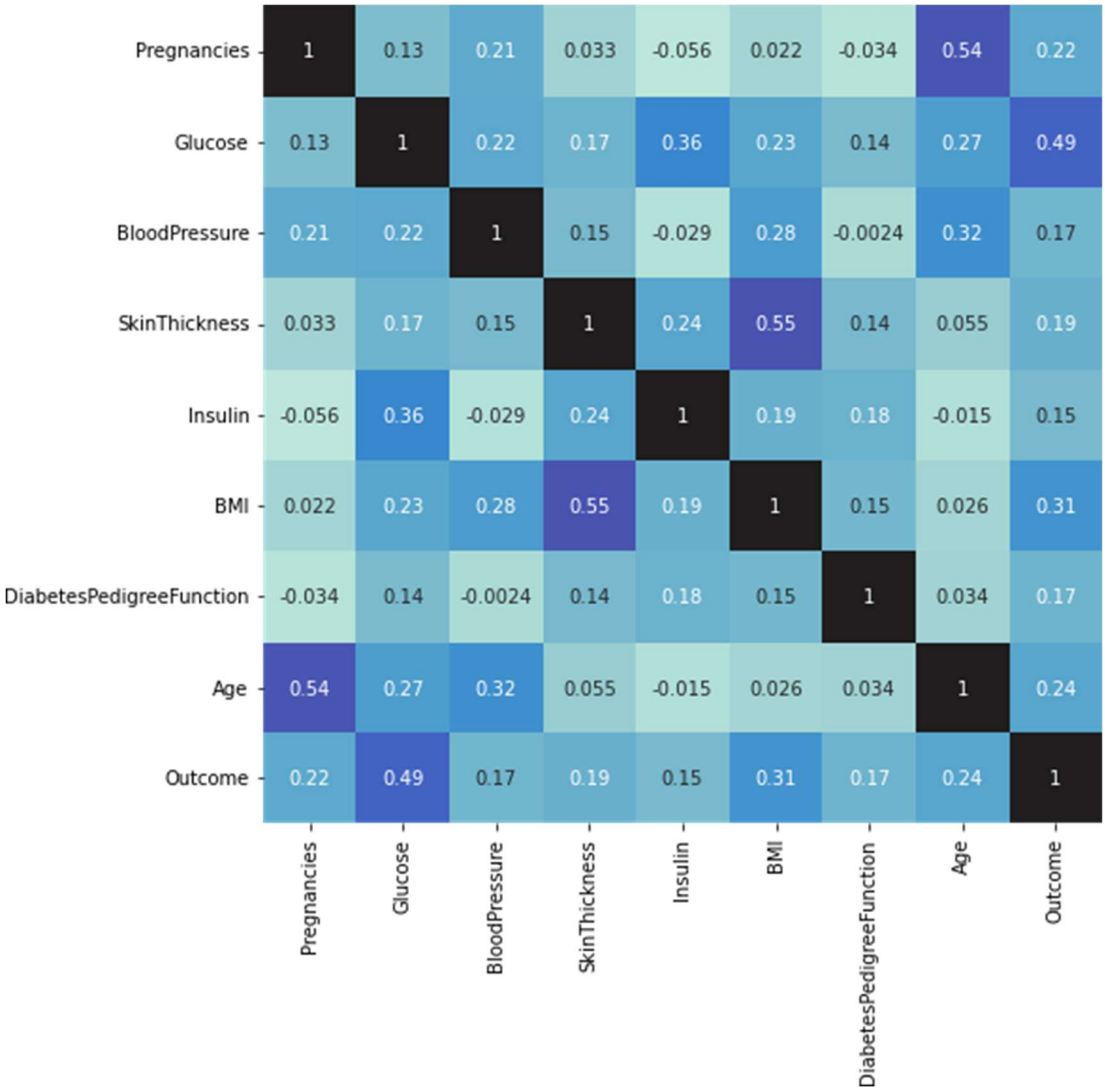
	Insulin	BMI	DiabetesPedigreeFunction	\
Pregnancies	-0.055697	0.021546	-0.033523	
Glucose	0.357573	0.231400	0.137327	
BloodPressure	-0.028721	0.281132	-0.002378	
SkinThickness	0.238188	0.546951	0.142977	
Insulin	1.000000	0.189022	0.178029	
BMI	0.189022	1.000000	0.153506	
DiabetesPedigreeFunction	0.178029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

```
plt.figure(figsize = (8,8))
sns.heatmap(df.corr(), annot = True, center = 1, cbar = False)
# sns.heatmap(df.corr(), annot = True, cmap = 'BuPu', cbar = False)
```

<AxesSubplot:>




```
df.corr().to_csv('correlation matrix.csv')
```

Predictive Analysis

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
```

```
confusion_matrix, precision_recall_curve
from sklearn.metrics import roc_curve, auc, roc_auc_score

X = df.drop(['Outcome'], axis = 1) y =
df['Outcome']
```

Random State 12 yielded the best output in every Category

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 12,
stratify = y)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(614, 8)
(154, 8)
(614,)
(154,)
```

Scaling

```
from sklearn.preprocessing import StandardScaler sc =

StandardScaler()


X_train = sc.fit_transform(X_train) X_test =
sc.transform(X_test)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

log = LogisticRegression()
log.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
y_pred = log.predict(X_test) y_pred_train =
log.predict(X_train)

print(accuracy_score(y_train, y_pred_train))
print(accuracy_score(y_test, y_pred))
```

```
0.7719869706840391
0.7727272727272727
```

```
print('Train', classification_report(y_train, y_pred_train)) print('Test',
classification_report(y_test, y_pred))
```

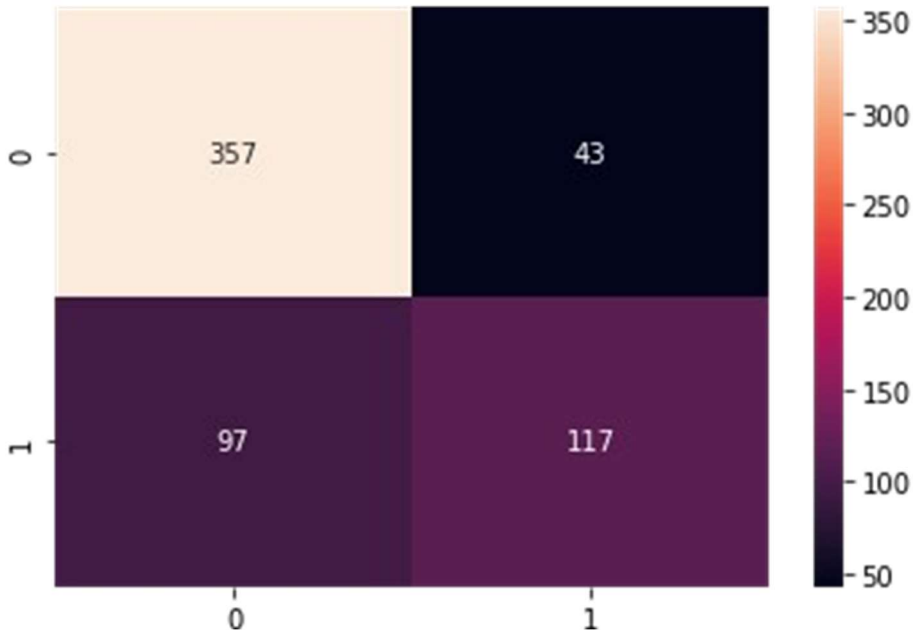
```
Train                    precision    recall  f1-score   support
```

0	0.79	0.89	0.84	400
1	0.73	0.55	0.63	214
accuracy			0.77	614
macro avg	0.76	0.72	0.73	614
weighted avg	0.77	0.77	0.76	614

Test	precision	recall	f1-score	support
0	0.81	0.85	0.83	100
1	0.69	0.63	0.66	54
accuracy			0.77	154
macro avg	0.75	0.74	0.74	154
weighted avg	0.77	0.77	0.77	154

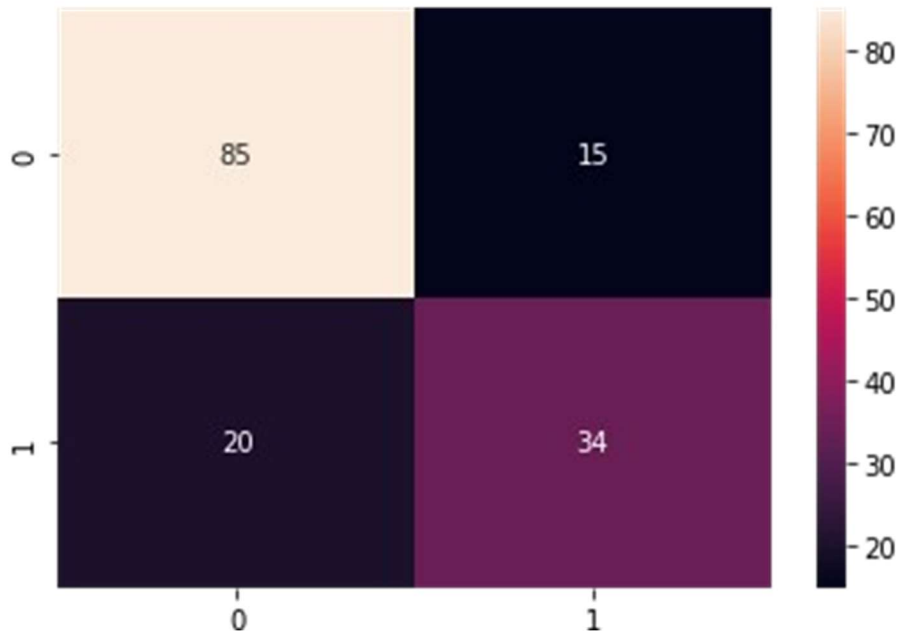
```
sns.heatmap(confusion_matrix(y_train, y_pred_train), annot = True, fmt
= 'g')
```

<AxesSubplot:>

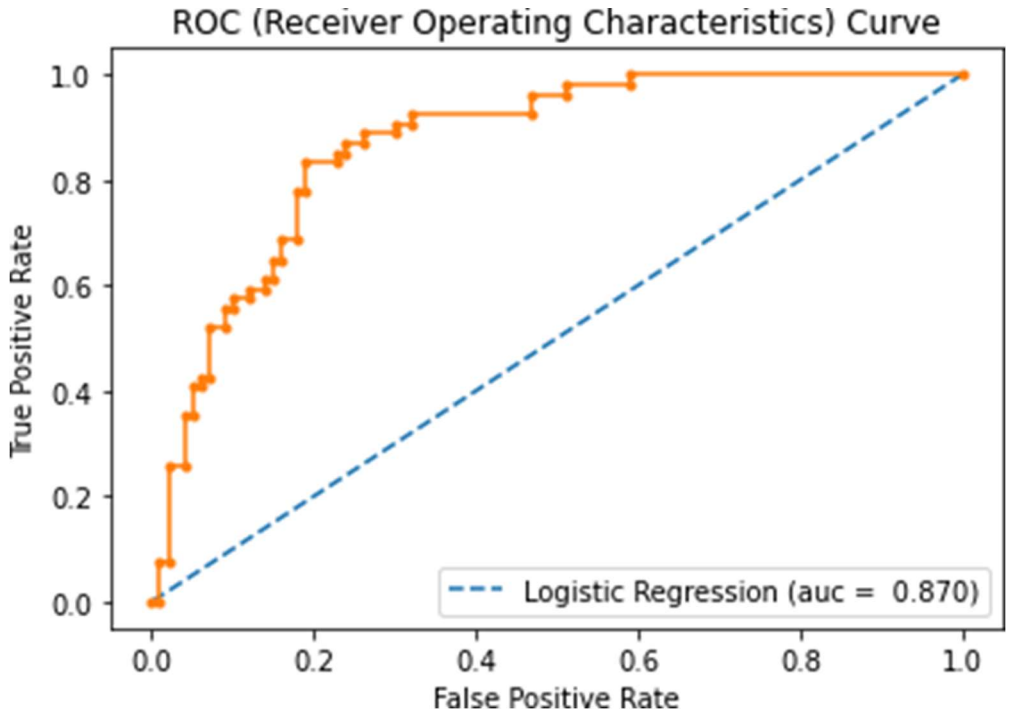



```
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True)
```

<AxesSubplot:>




```
probs_log = log.predict_proba(X_test) probs_log =
probs_log[:, 1]

auc_log = roc_auc_score(y_test, probs_log) print('AUC:
%.3f' %auc_log)
fpr, tpr, thresholds = roc_curve(y_test, probs_log)

plt.plot([0, 1], [0, 1], linestyle='--', label = 'Logistic Regression (auc =          %0.3f)' % auc_log)
plt.plot(fpr, tpr, marker='.') plt.legend()

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve"); AUC: 0.870
```



KNN

from sklearn.neighbors import KNeighborsClassifier

- n_neighbors 13 yielded the best possible results,
- 'brute' algorithm gave the best results in my testing
- in order to prevent overfitting/underfitting the following set was chosen

```
knn = KNeighborsClassifier(n_neighbors = 13, metric = 'euclidean', algorithm = 'brute')
knn.fit(X_train, y_train)
```

KNeighborsClassifier(algorithm='brute', metric='euclidean', n_neighbors=13)

```
knn_pred = knn.predict(X_test) knn_train =
knn.predict(X_train)
```

```
print(accuracy_score(y_train, knn_train))
print(accuracy_score(y_test, knn_pred))
```

0.7785016286644951
0.7727272727272727

```
print('Train', classification_report(y_train, knn_train)) print('Test'
,classification_report(y_test, knn_pred))
```

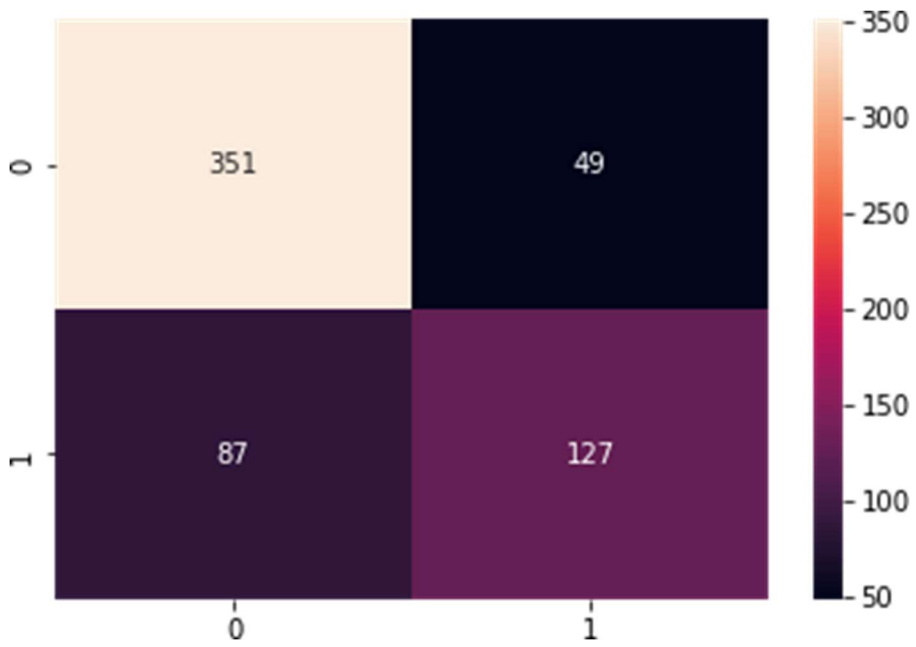
Train	precision	recall	f1-score	support 0
	0.80	0.88	0.84	400

1	0.72	0.59	0.65	214
accuracy			0.78	614
macro avg	0.76	0.74	0.74	614
weighted avg	0.77	0.78	0.77	614

Test	precision	recall	f1-score	support
0	0.80	0.87	0.83	100
1	0.71	0.59	0.65	54
accuracy			0.77	154
macro avg	0.75	0.73	0.74	154
weighted avg	0.77	0.77	0.77	154

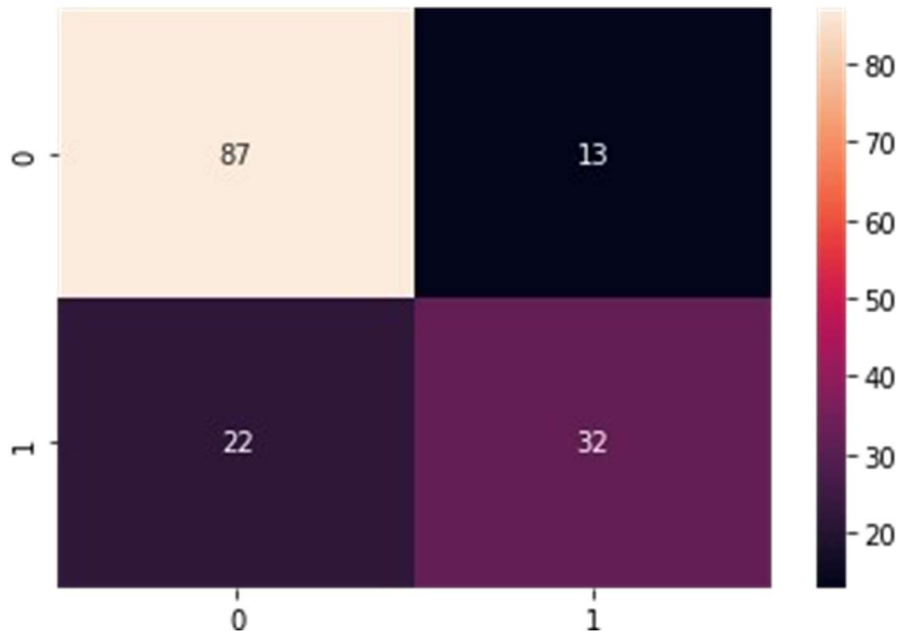
```
sns.heatmap(confusion_matrix(y_train, knn_train), annot = True, fmt = 'g')
```

<AxesSubplot:>

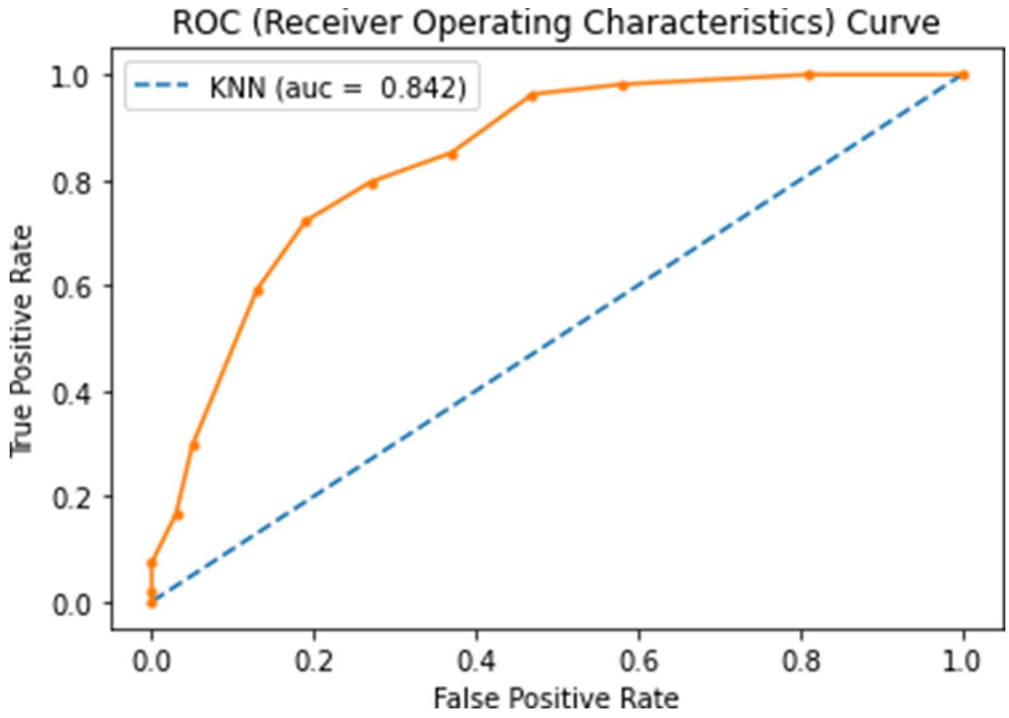



```
sns.heatmap(confusion_matrix(y_test, knn_pred), annot = True, fmt = 'g')
```

<AxesSubplot:>




```
probs_knn = knn.predict_proba(X_test) probs_knn
= probs_knn[:, 1]

auc_knn = roc_auc_score(y_test, probs_knn) print('AUC:
%.3f' % auc_knn)
fpr, tpr, thresholds = roc_curve(y_test, probs_knn)

plt.plot([0, 1], [0, 1], linestyle='--', label = 'KNN (auc =                    %0.3f)'
% auc_knn)
plt.plot(fpr, tpr, marker='.') plt.legend()

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve"); AUC: 0.842
```



SVM

from sklearn.svm import SVC

- kernel = linear proved to better suited than default rbf, poly and signoid
- probability is True for ROC-AUC Graph to work

model = SVC(kernel = 'linear', degree = 5, probability = True) model.fit(X_train, y_train)

SVC(degree=5, kernel='linear', probability=True) model_pred =

model.predict(X_test)
model_train = model.predict(X_train)

print(accuracy_score(y_train, model_train))
print(accuracy_score(y_test, model_pred))

0.7703583061889251
0.7792207792207793

print('Train', classification_report(y_train, model_train)) print('Test' ,classification_report(y_test, model_pred))

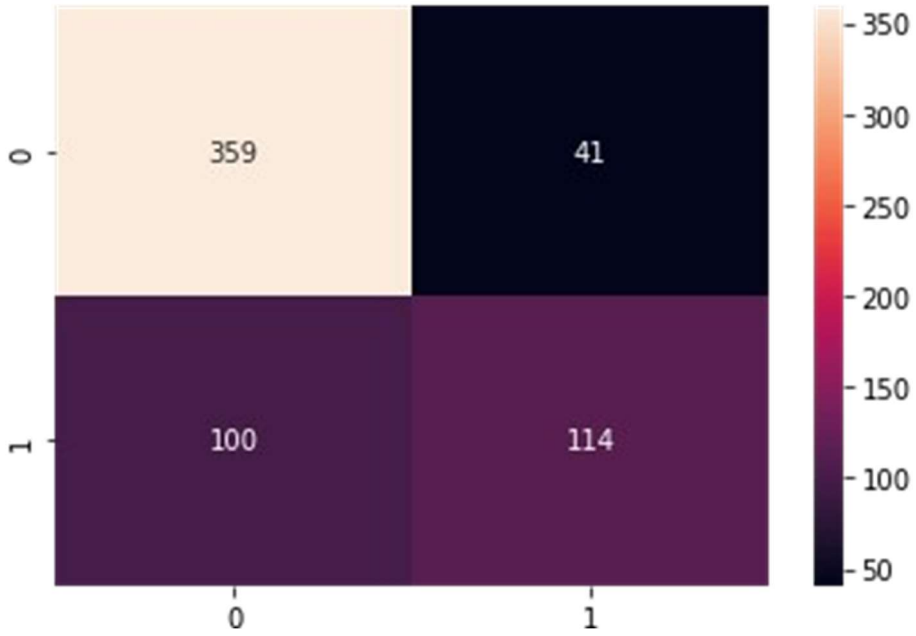
Train		precision	recall	f1-score	support
	0	0.78	0.90	0.84	400
	1	0.74	0.53	0.62	214
accuracy				0.77	614

macro avg	0.76	0.72	0.73	614
weighted avg	0.77	0.77	0.76	614

Test	precision	recall	f1-score	support
0	0.81	0.86	0.83	100
1	0.71	0.63	0.67	54
accuracy			0.78	154
macro avg	0.76	0.74	0.75	154
weighted avg	0.78	0.78	0.78	154

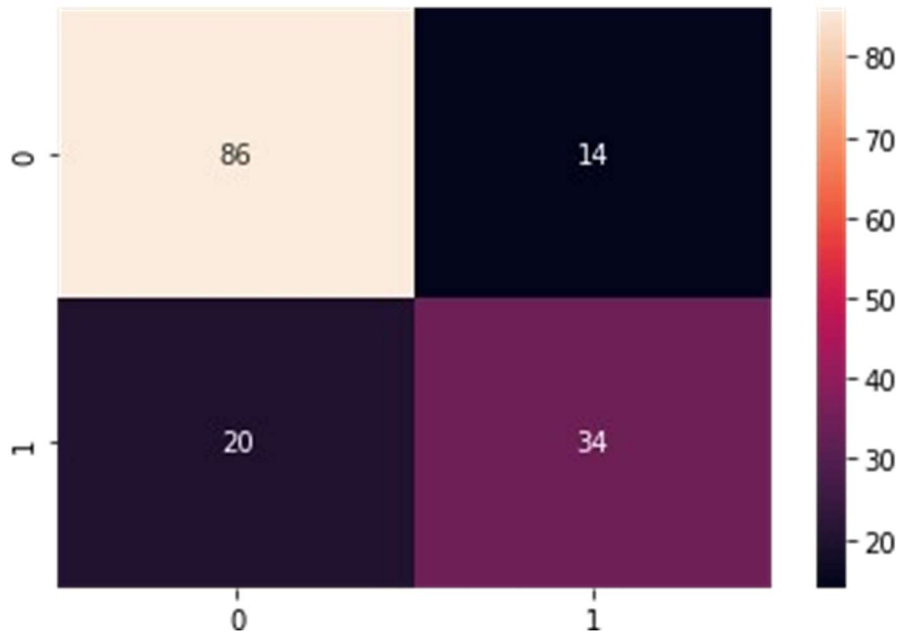
```
sns.heatmap(confusion_matrix(y_train, model_train), annot = True, fmt
= 'g')
```

<AxesSubplot:>

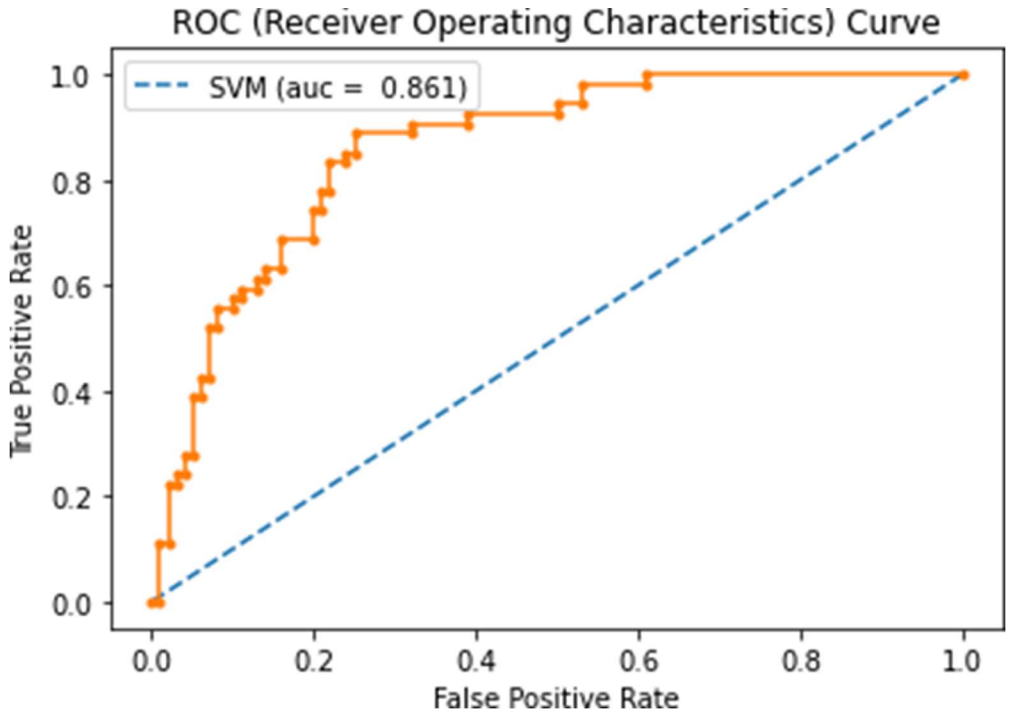



```
sns.heatmap(confusion_matrix(y_test, model_pred), annot = True)
```

<AxesSubplot:>




```
probs_model = model.predict_proba(X_test)
probs_model = probs_model[:, 1]

auc_model = roc_auc_score(y_test, probs_model) print('AUC:
%.3f' % auc_model)
fpr, tpr, thresholds = roc_curve(y_test, probs_model)

plt.plot([0, 1], [0, 1], linestyle='--', label = 'SVM (auc =                    %0.3f)'
% auc_model)
plt.plot(fpr, tpr, marker='.') plt.legend()

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve"); AUC: 0.861
```



Decision Tree

from sklearn.tree import DecisionTreeClassifier

- Entropy gave the best Overfitting/Underfitting Results

```
dt = DecisionTreeClassifier(criterion = 'entropy', max_depth = 2, random_state = 3)
dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=2, random_state=3)
```

```
dt_pred = dt.predict(X_test) dt_train =
dt.predict(X_train)
```

```
print('Train :', accuracy_score(y_train, dt_train)) print('Test :'
,accuracy_score(y_test, dt_pred))
```

```
Train : 0.7719869706840391
Test : 0.7662337662337663
```

```
print('Train', classification_report(y_train, dt_train)) print('Test'
,classification_report(y_test, dt_pred))
```

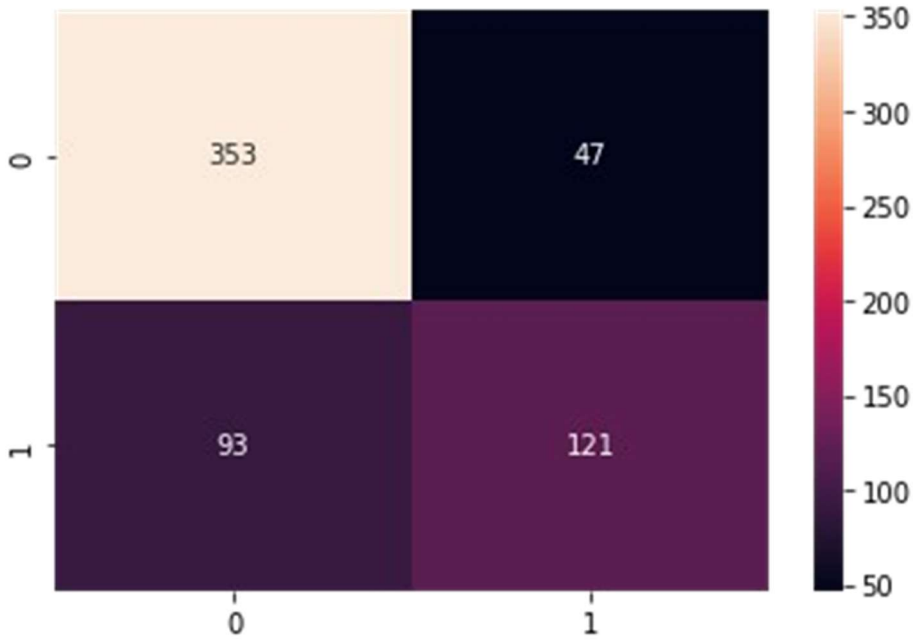
Train		precision	recall	f1-score	support
	0	0.79	0.88	0.83	400
	1	0.72	0.57	0.63	214
	accuracy			0.77	614

macro avg	0.76	0.72	0.73	614
weighted avg	0.77	0.77	0.76	614

Test	precision	recall	f1-score	support
0	0.78	0.89	0.83	100
1	0.72	0.54	0.62	54
accuracy			0.77	154
macro avg	0.75	0.71	0.72	154
weighted avg	0.76	0.77	0.76	154

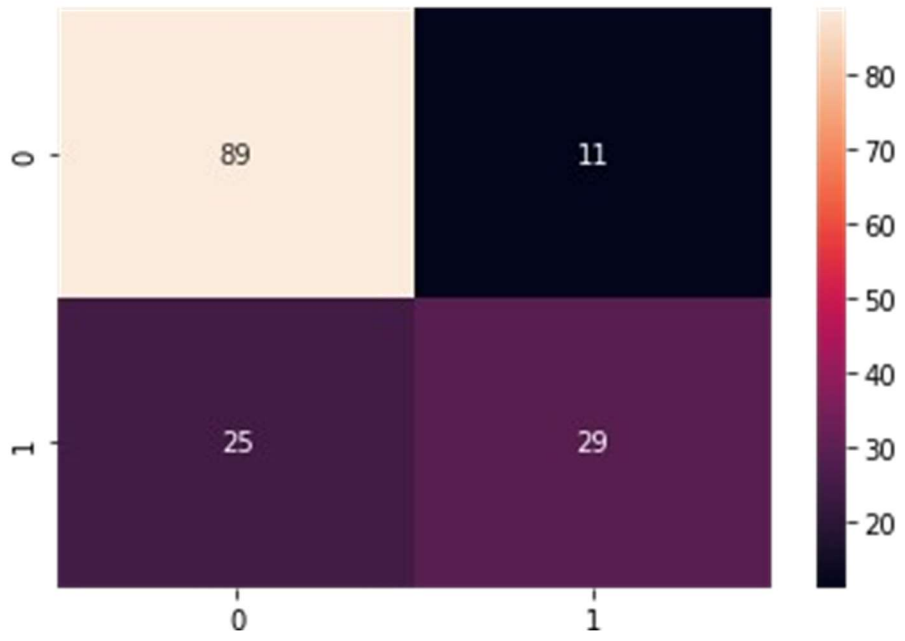
```
sns.heatmap(confusion_matrix(y_train, dt_train), annot = True, fmt = 'g')
```

<AxesSubplot:>

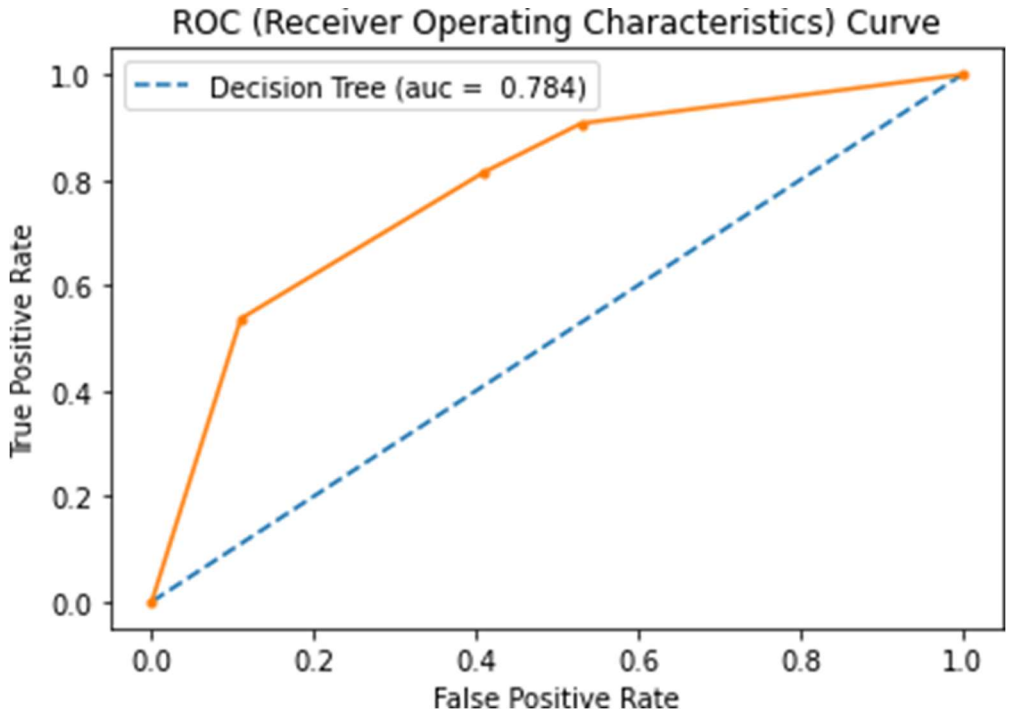



```
sns.heatmap(confusion_matrix(y_test, dt_pred), annot = True, fmt = 'g')
```

<AxesSubplot:>




```
probs_dt = dt.predict_proba(X_test) probs_dt =
probs_dt[:, 1]

auc_dt = roc_auc_score(y_test, probs_dt) print('AUC:
%.3f' % auc_dt)
fpr, tpr, thresholds = roc_curve(y_test, probs_dt)

plt.plot([0, 1], [0, 1], linestyle='--', label = 'Decision Tree (auc =
%0.3f)' % auc_dt) plt.plot(fpr, tpr,
marker='.')

plt.legend()

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve"); AUC: 0.784
```



Random Forest

from sklearn.ensemble import RandomForestClassifier

- Number of trees is being set to 200
- Entropy proved to be better suited than Gini & Log_loss

```
rf = RandomForestClassifier(n_estimators = 200, criterion = 'entropy', random_state = 13,
max_depth = 2, max_features = 'log2') rf.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', max_depth=2,
max_features='log2',
                                    n_estimators=200, random_state=13)
```

```
rf_pred = rf.predict(X_test) rf_train =
rf.predict(X_train)
```

```
print('Train :', accuracy_score(y_train, rf_train)) print('Test :'
,accuracy_score(y_test, rf_pred))
```

```
Train : 0.7768729641693811
Test : 0.7662337662337663
```

```
print('Train', classification_report(y_train, rf_train)) print('Test'
,classification_report(y_test, rf_pred))
```

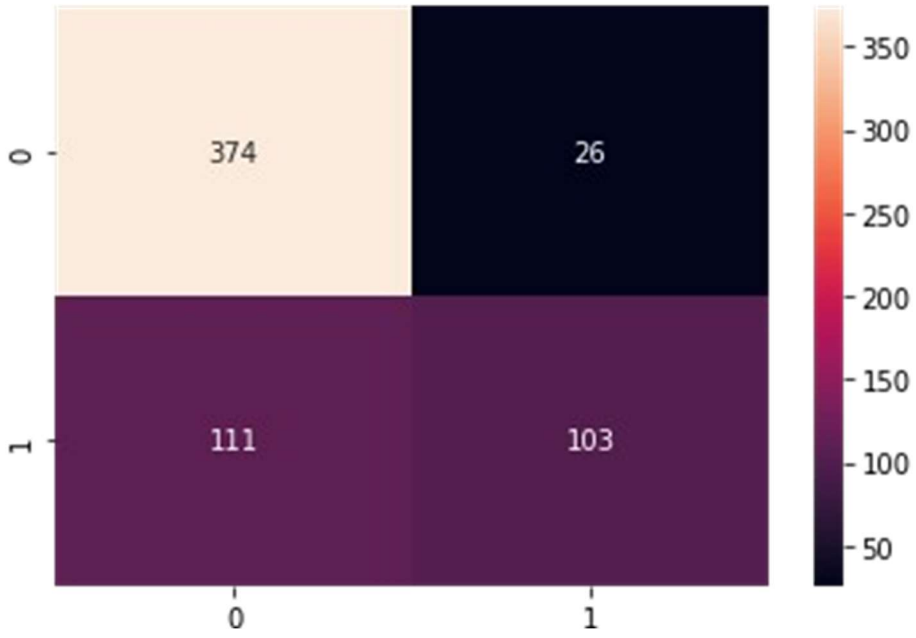
Train	precision	recall	f1-score	support 0
	0.77	0.94	0.85	400

1	0.80	0.48	0.60	214
accuracy			0.78	614
macro avg	0.78	0.71	0.72	614
weighted avg	0.78	0.78	0.76	614

Test	precision	recall	f1-score	support
0	0.76	0.93	0.84	100
1	0.78	0.46	0.58	54
accuracy			0.77	154
macro avg	0.77	0.70	0.71	154
weighted avg	0.77	0.77	0.75	154

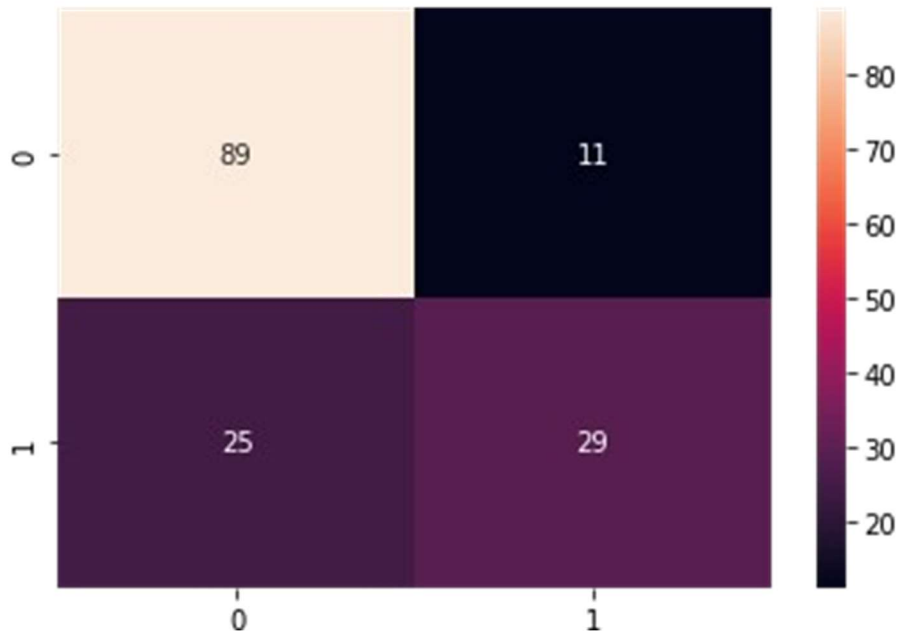
```
sns.heatmap(confusion_matrix(y_train, rf_train), annot = True, fmt = 'g')
```

<AxesSubplot:>

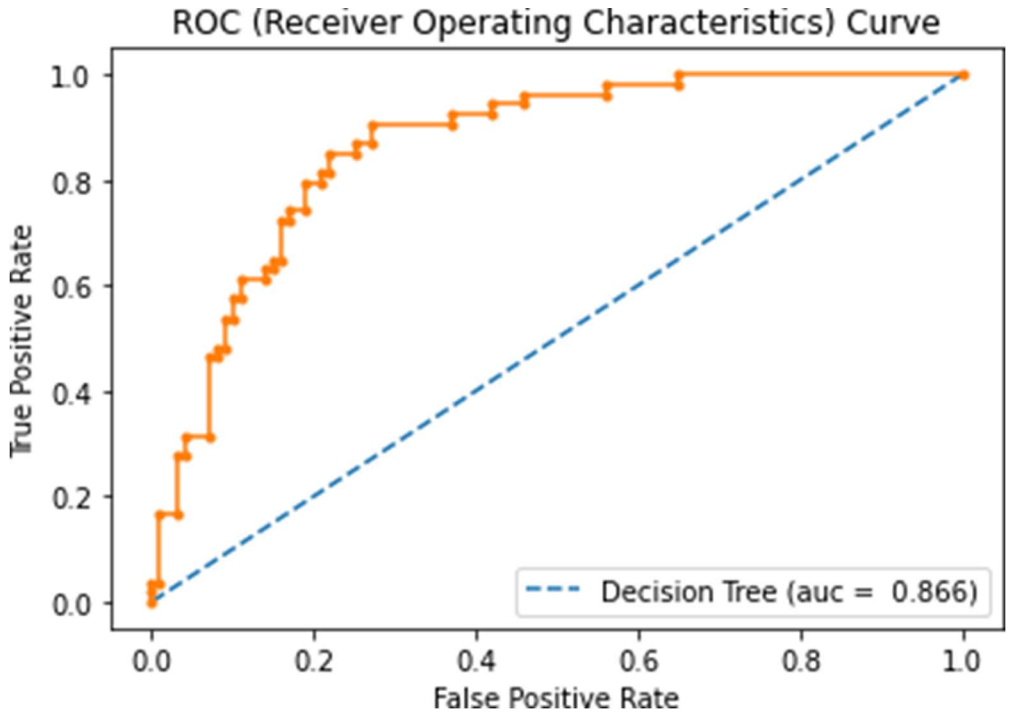



```
sns.heatmap(confusion_matrix(y_test, dt_pred), annot = True, fmt = 'g')
```

<AxesSubplot:>

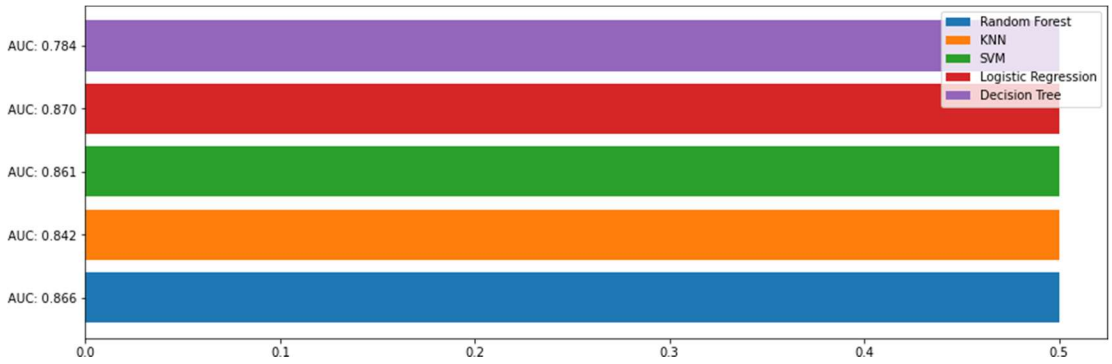



```
prob_rf = rf.predict_proba(X_test) prob_rf =
prob_rf[:, 1]

auc_rf = roc_auc_score(y_test, prob_rf) print('AUC:
%.3f' % auc_rf)
fpr, tpr, thresholds = roc_curve(y_test, prob_rf)

plt.plot([0, 1], [0, 1], linestyle='--', label = 'Decision Tree (auc =
%0.3f)' % auc_rf) plt.plot(fpr, tpr,
marker='.')

plt.legend()

plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC (Receiver Operating Characteristics) Curve"); AUC: 0.866
```



Best ROC-AUC Performance

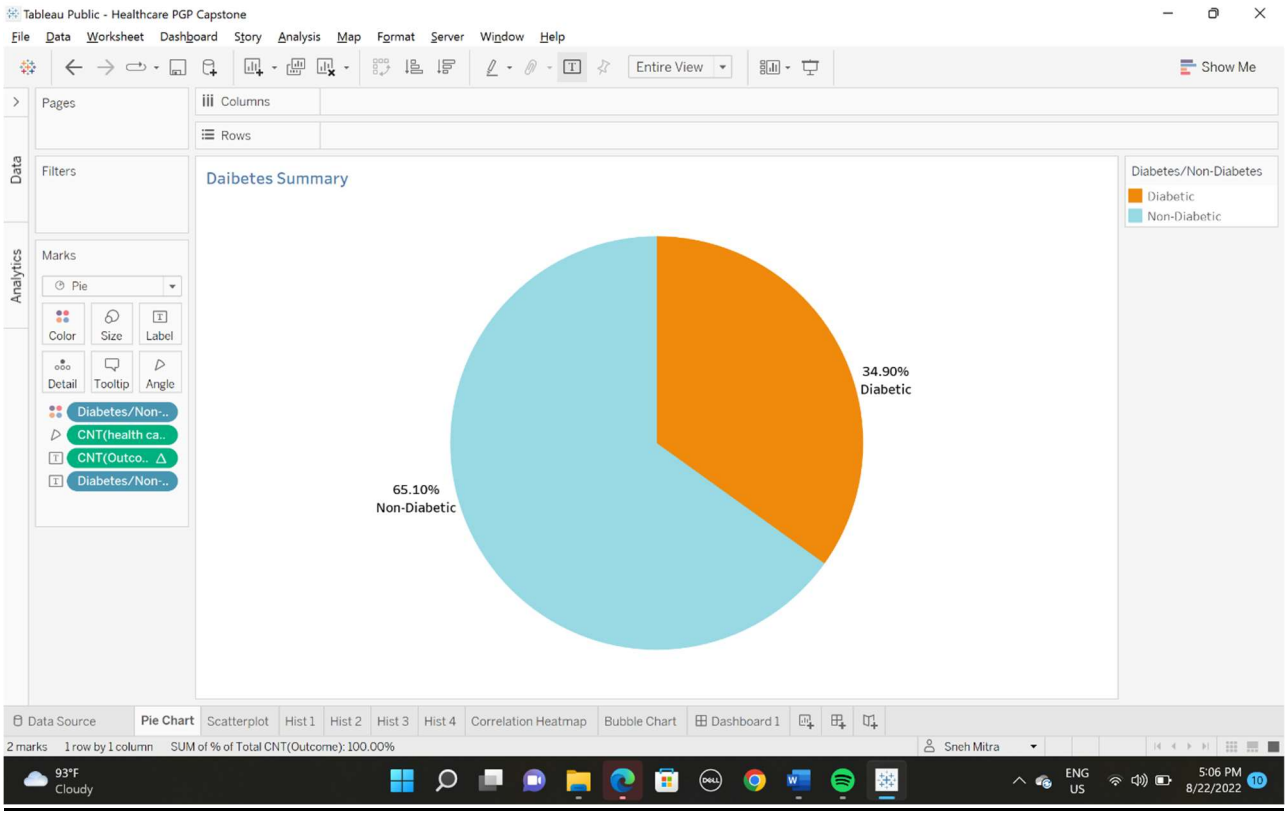
```
randomforest = 'AUC: %.3f' % auc_rf
decisiontree = 'AUC: %.3f' % auc_dt svm = 'AUC:
%.3f' % auc_model
KNN = 'AUC: %.3f' % auc_knn Log =
'AUC: %.3f' % auc_log

plt.figure(figsize = (15,5))

plt.barh(randomforest, label = 'Random Forest', width = 0.5) plt.barh(KNN, label =
'KNN', width = 0.5)
plt.barh(svm, label = 'SVM', width = 0.5)
plt.barh(Log, label = 'Logistic Regression', width = 0.5) plt.barh(decisiontree, label =
'Decision Tree', width = 0.5)

plt.legend()

<matplotlib.legend.Legend at 0x20bad1b29d0>
```



As per the results of ROC-AUC Curve, Logistic Regression and Support Vector Machine both perform better than K-Nearest Neighbor. Random Forest Classifier and Decision Tree Classifier both failed to provide the desired output.

TABLEAU

TABLEAU LINK:
https://public.tableau.com/shared/WSBH2B3ND?:display_count=n&:origin=viz_share_link



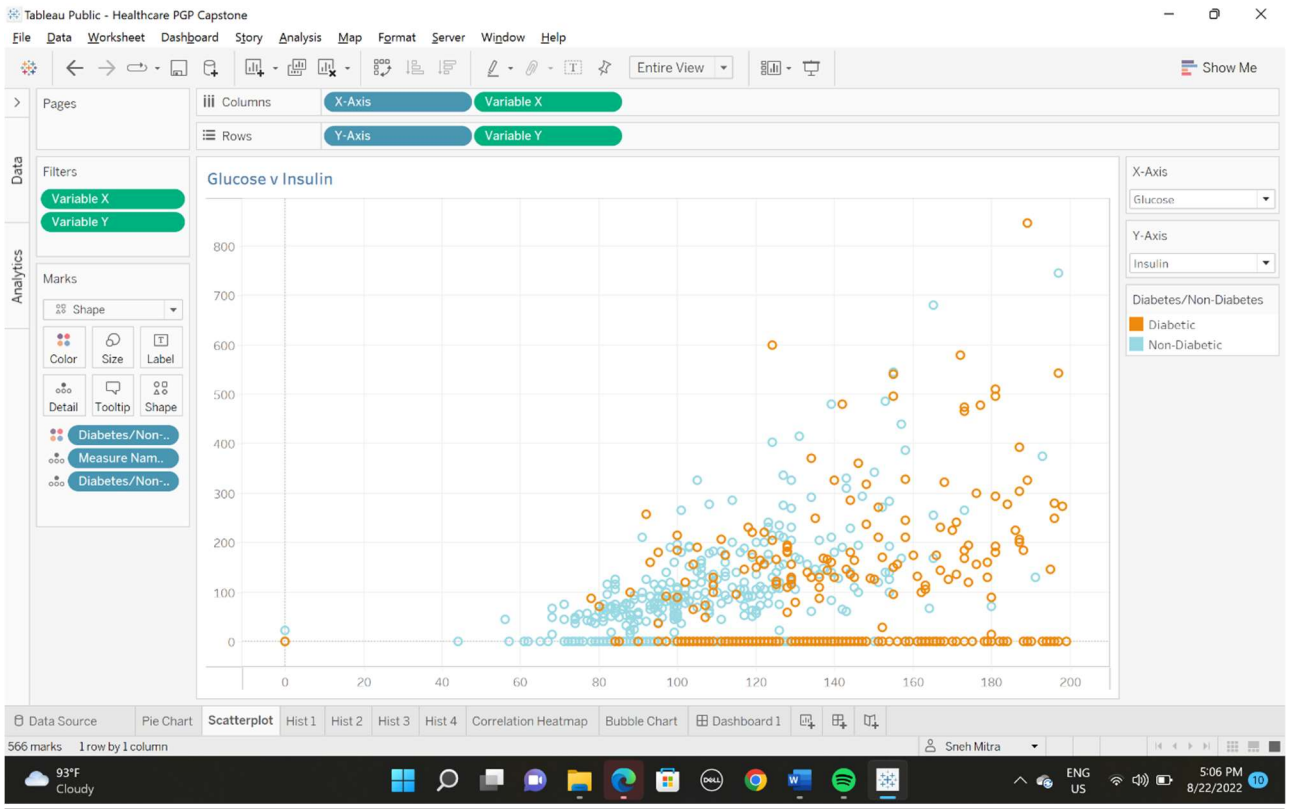


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns X-Axis Variable X
Rows Y-Axis Variable Y
Filters
Variable X
Variable Y
Marks
Shape
Color Size Label
Detail Tooltip Shape
Diabetes/Non-..
Measure Nam..
Diabetes/Non-..
Glucose v Insulin
800 700 600 500 400 300 200 100 0
0 20 40 60 80 100 120 140 160 180 200
X-Axis
Glucose
Y-Axis
Insulin
Diabetes/Non-Diabetes
Diabetic
Non-Diabetic
Data Source Pie Chart Scatterplot Hist 1 Hist 2 Hist 3 Hist 4 Correlation Heatmap Bubble Chart Dashboard 1
566 marks 1 row by 1 column
Sneh Mitra
93°F Cloudy
ENG US
5:06 PM 8/22/2022

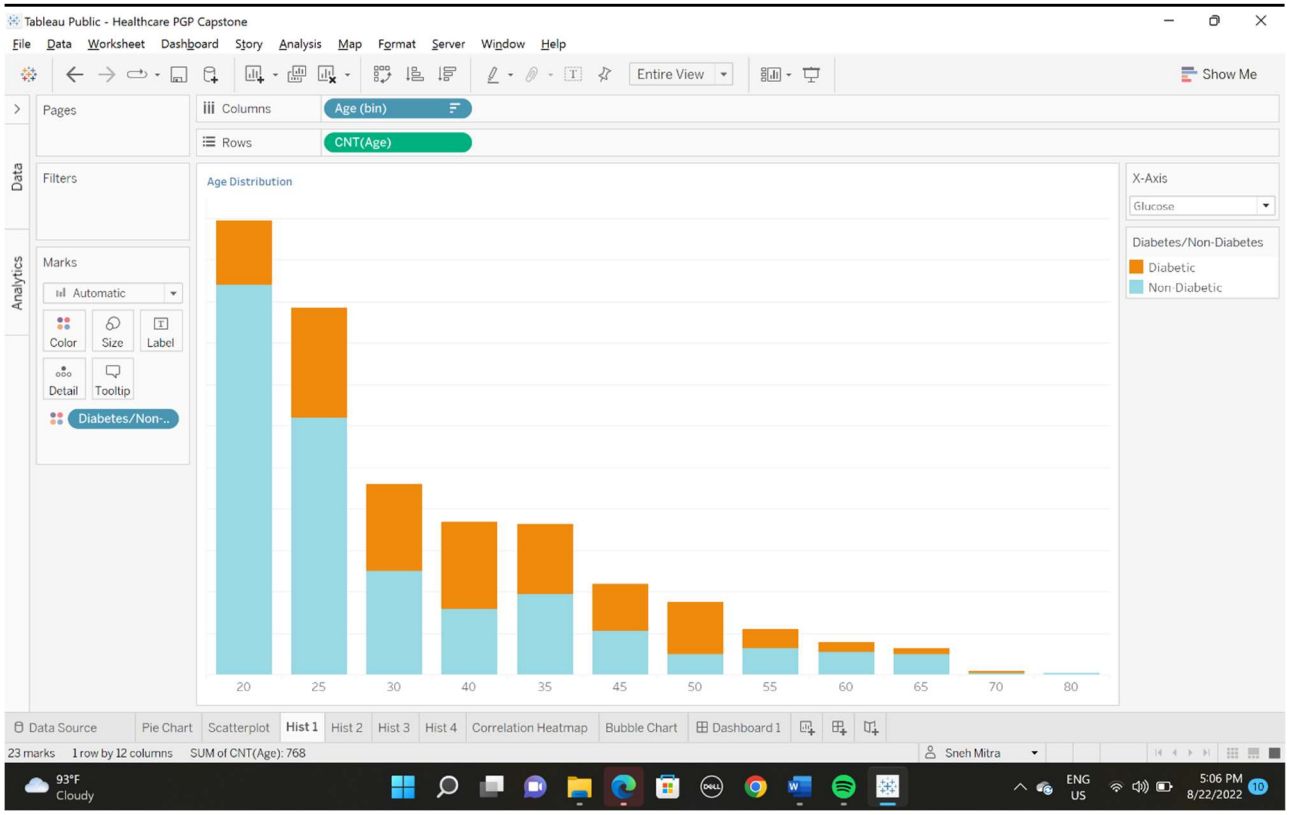


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns Age (bin)
Rows CNT(Age)
Filters
Marks
Automatic
Color Size Label
Detail Tooltip
Diabetes/Non-..
Age Distribution
20 25 30 40 35 45 50 55 60 65 70 80
X-Axis
Glucose
Diabetes/Non-Diabetes
Diabetic
Non-Diabetic
Data Source Pie Chart Scatterplot Hist 1 Hist 2 Hist 3 Hist 4 Correlation Heatmap Bubble Chart Dashboard 1
23 marks 1 row by 12 columns SUM of CNT(Age): 768
Sneh Mitra
93°F Cloudy
ENG US
5:06 PM 8/22/2022

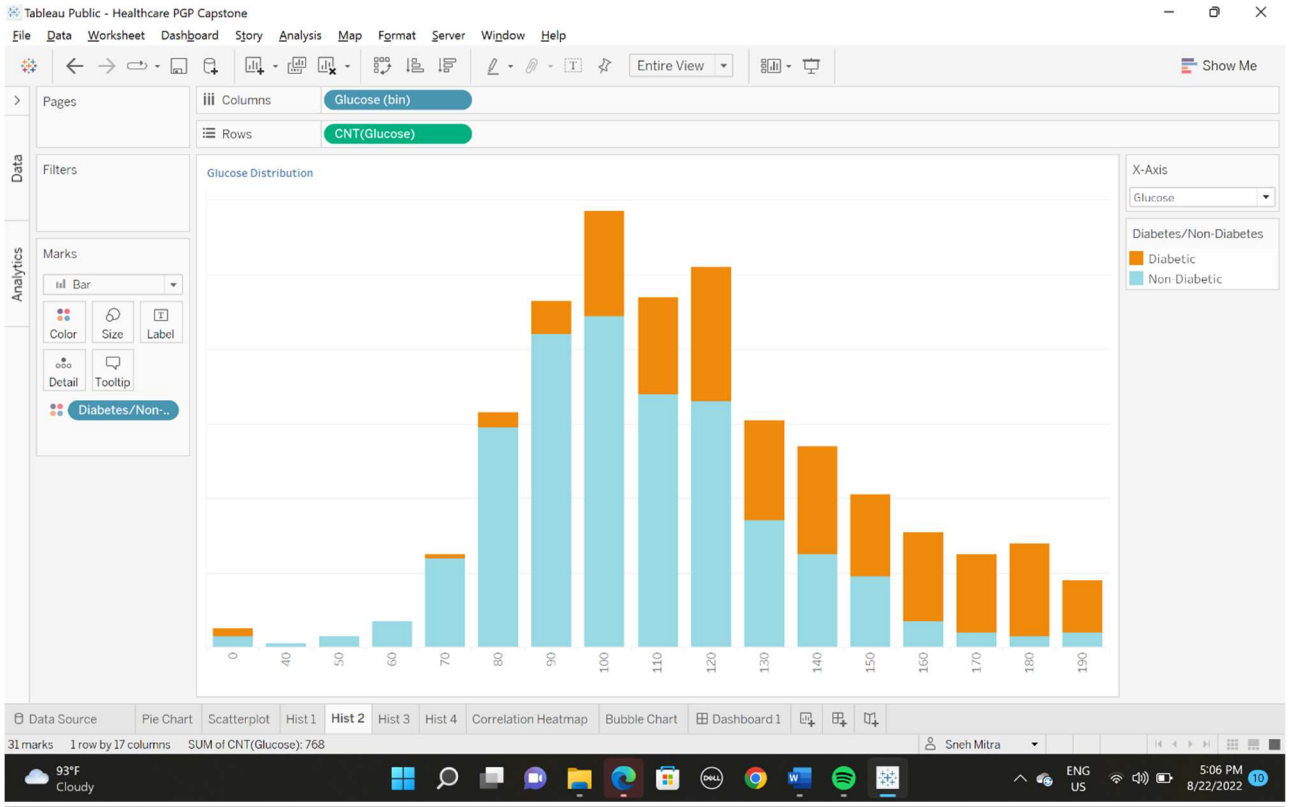


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns
Glucose (bin)
Rows
CNT(Glucose)
Filters
Glucose Distribution
X-Axis
Glucose
Diabetes/Non-Diabetes
Diabetic
Non-Diabetic
Marks
Bar
Color
Size
Label
Detail
Tooltip
Diabetes/Non-..
0
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
31 marks
1 row by 17 columns
SUM of CNT(Glucose): 768
Sneh Mitra
93°F
Cloudy
ENG
US
5:06 PM
8/22/2022

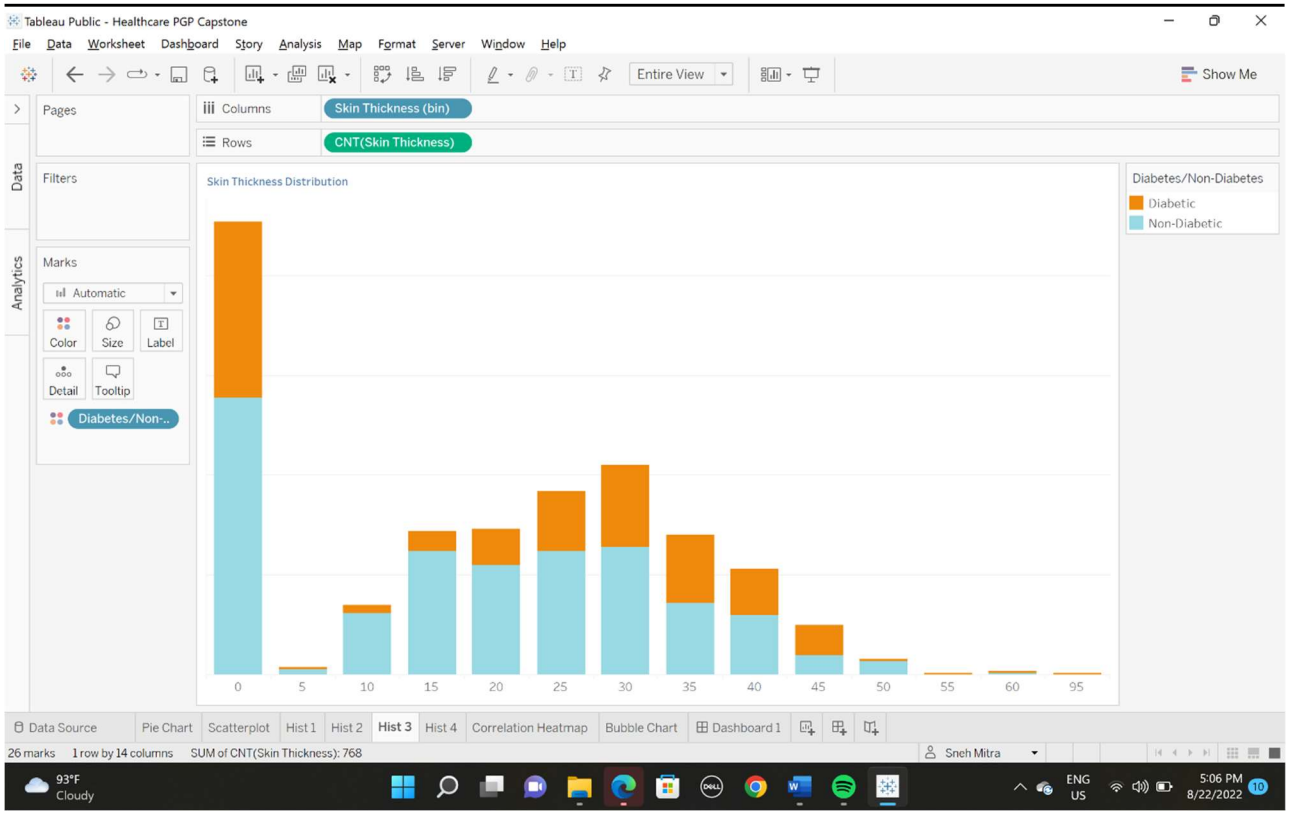


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns
Skin Thickness (bin)
Rows
CNT(Skin Thickness)
Filters
Skin Thickness Distribution
Diabetes/Non-Diabetes
Diabetic
Non-Diabetic
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Diabetes/Non-..
0
5
10
15
20
25
30
35
40
45
50
55
60
95
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
26 marks
1 row by 14 columns
SUM of CNT(Skin Thickness): 768
Sneh Mitra
93°F
Cloudy
ENG
US
5:06 PM
8/22/2022

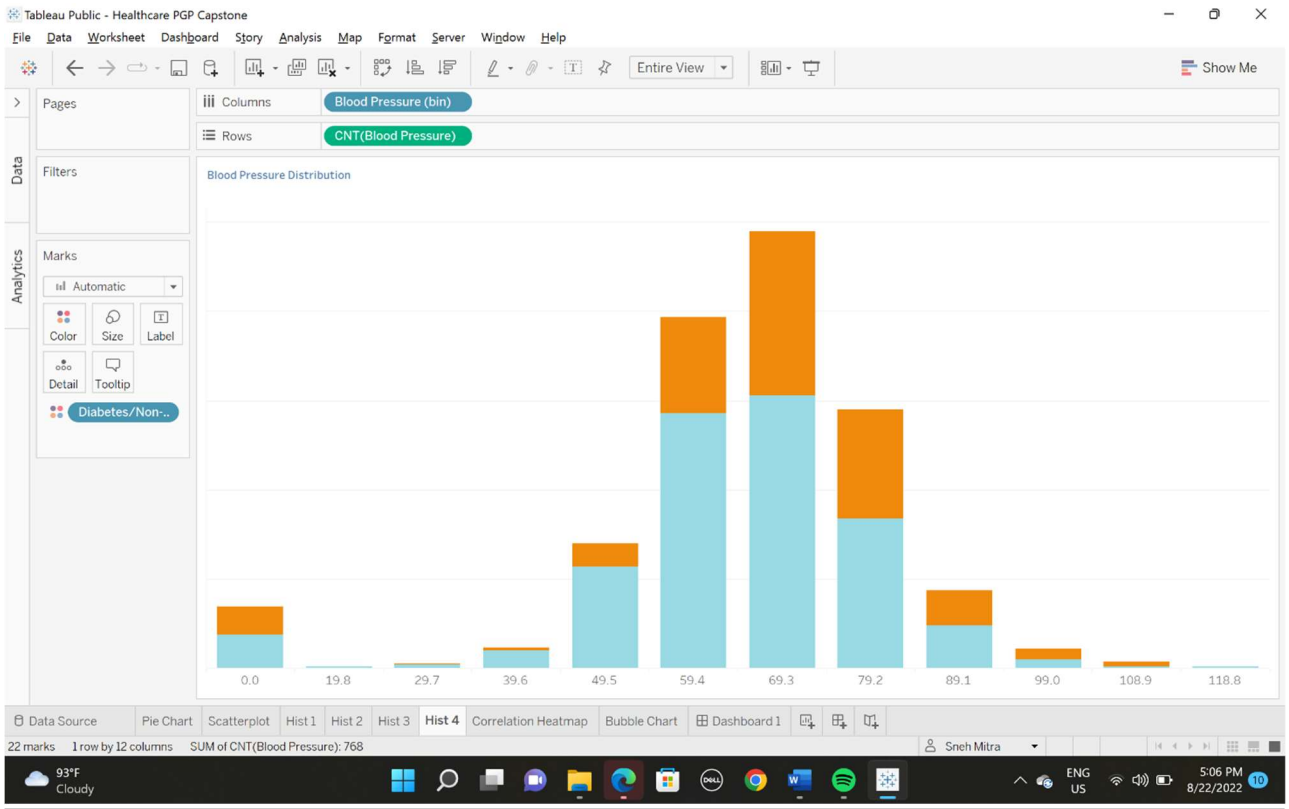


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns
Blood Pressure (bin)
Rows
CNT(Blood Pressure)
Filters
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Diabetes/Non-..
Blood Pressure Distribution
0.0
19.8
29.7
39.6
49.5
59.4
69.3
79.2
89.1
99.0
108.9
118.8
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
22 marks
1 row by 12 columns
SUM of CNT(Blood Pressure): 768
Sneh Mitra
93°F
Cloudy
ENG
US
5:06 PM
8/22/2022

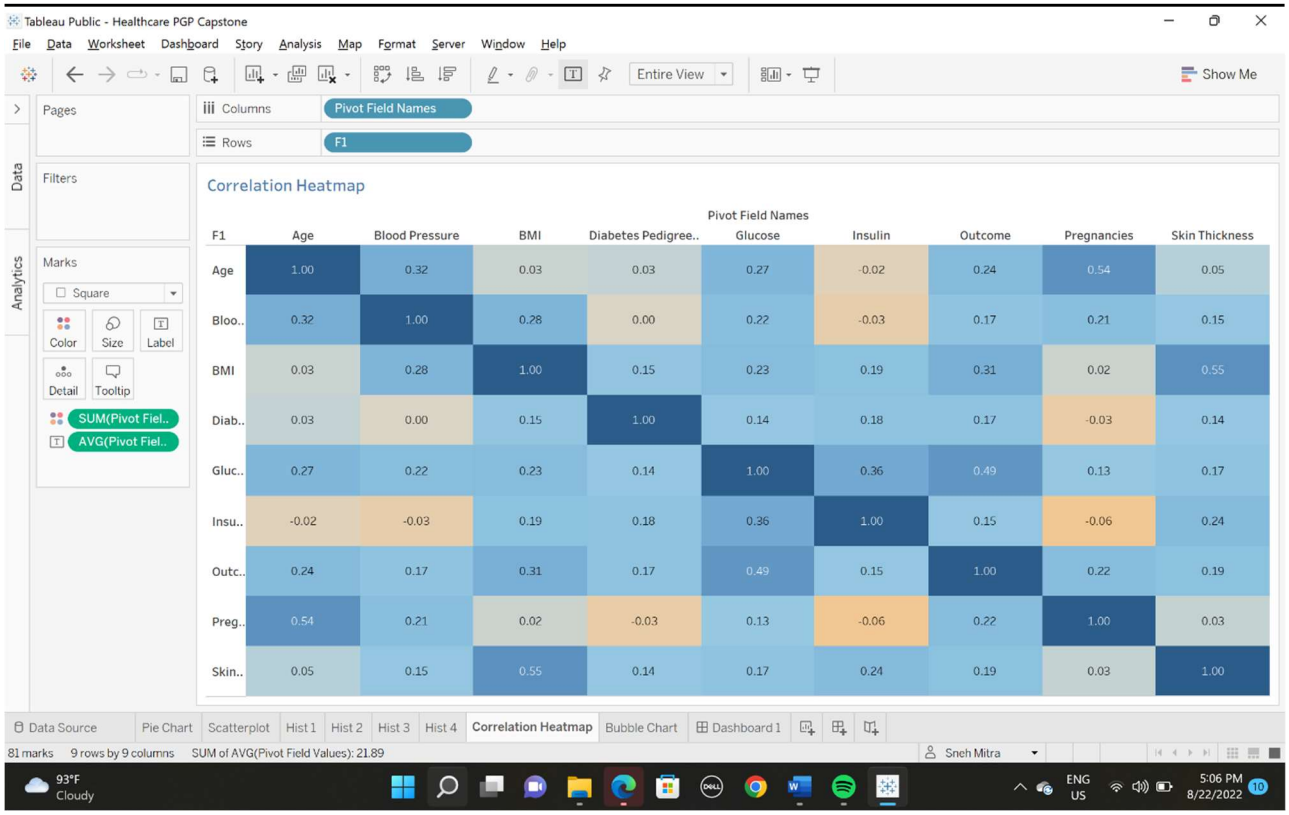


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns
Pivot Field Names
Rows
F1
Filters
Marks
Square
Color
Size
Label
Detail
Tooltip
SUM(Pivot Fiel..
AVG(Pivot Fiel..
Correlation Heatmap
Pivot Field Names
F1 Age Blood Pressure BMI Diabetes Pedigree.. Glucose Insulin Outcome Pregnancies Skin Thickness
Age 1.00 0.32 0.03 0.03 0.27 -0.02 0.24 0.54 0.05
Bloo.. 0.32 1.00 0.28 0.00 0.22 -0.03 0.17 0.21 0.15
BMI 0.03 0.28 1.00 0.15 0.23 0.19 0.31 0.02 0.55
Diab.. 0.03 0.00 0.15 1.00 0.14 0.18 0.17 -0.03 0.14
Gluc.. 0.27 0.22 0.23 0.14 1.00 0.36 0.49 0.13 0.17
Insu.. -0.02 -0.03 0.19 0.18 0.36 1.00 0.15 -0.06 0.24
Outc.. 0.24 0.17 0.31 0.17 0.49 0.15 1.00 0.22 0.19
Preg.. 0.54 0.21 0.02 -0.03 0.13 -0.06 0.22 1.00 0.03
Skin.. 0.05 0.15 0.55 0.14 0.17 0.24 0.19 0.03 1.00
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
81 marks
9 rows by 9 columns
SUM of AVG(Pivot Field Values): 21.89
Sneh Mitra
93°F
Cloudy
ENG
US
5:06 PM
8/22/2022

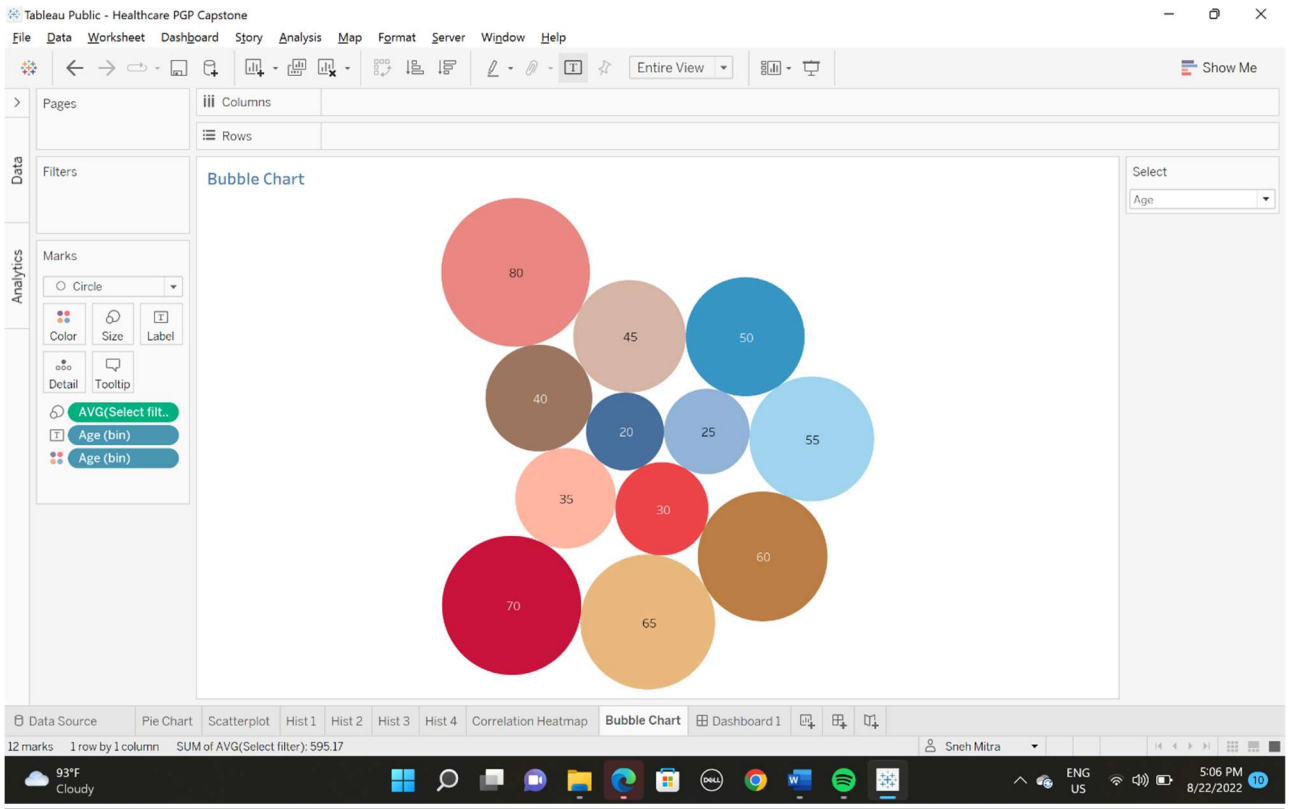


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Entire View
Show Me
Pages
Columns
Rows
Filters
Marks
Circle
Color
Size
Label
Detail
Tooltip
AVG(Select filt..
Age (bin)
Age (bin)
Bubble Chart
Select
Age
80
45
50
40
20
25
55
35
30
60
70
65
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
12 marks
1 row by 1 column
SUM of AVG(Select filter): 595.17
Sneh Mitra
93°F
Cloudy
ENG
US
5:06 PM
8/22/2022

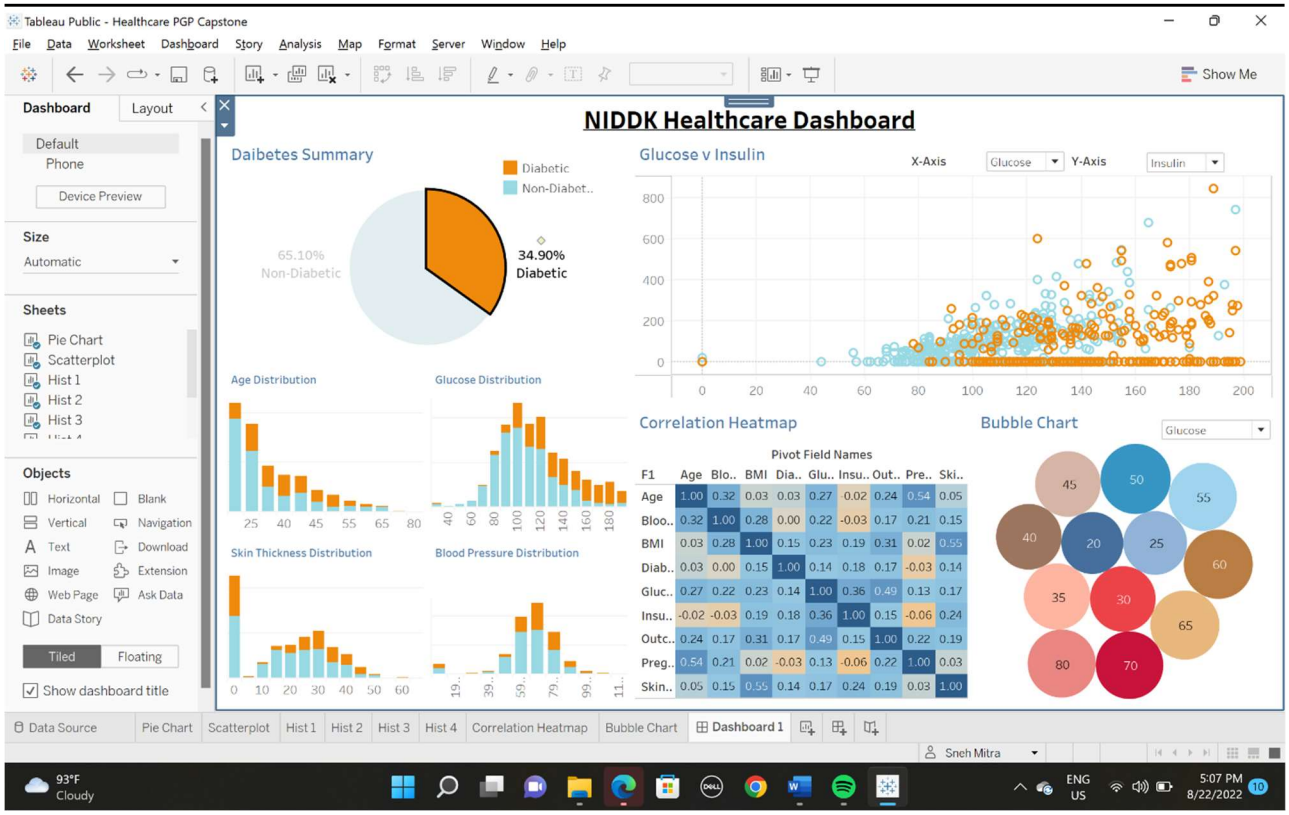


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Show Me
Dashboard
Layout
Default
Phone
Device Preview
Size
Automatic
Sheets
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Objects
Horizontal
Blank
Vertical
Navigation
Text
Download
Image
Extension
Web Page
Ask Data
Data Story
Tiled
Floating
Show dashboard title
NIDDK Healthcare Dashboard
Daibetes Summary
Diabetic
Non-Diabet..
65.10%
Non-Diabetic
34.90%
Diabetic
Glucose v Insulin
X-Axis
Glucose
Y-Axis
Insulin
Age Distribution
Glucose Distribution
Skin Thickness Distribution
Blood Pressure Distribution
Correlation Heatmap
Pivot Field Names
Bubble Chart
Glucose
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
Sneh Mitra
93°F
Cloudy
ENG
US
5:07 PM
8/22/2022

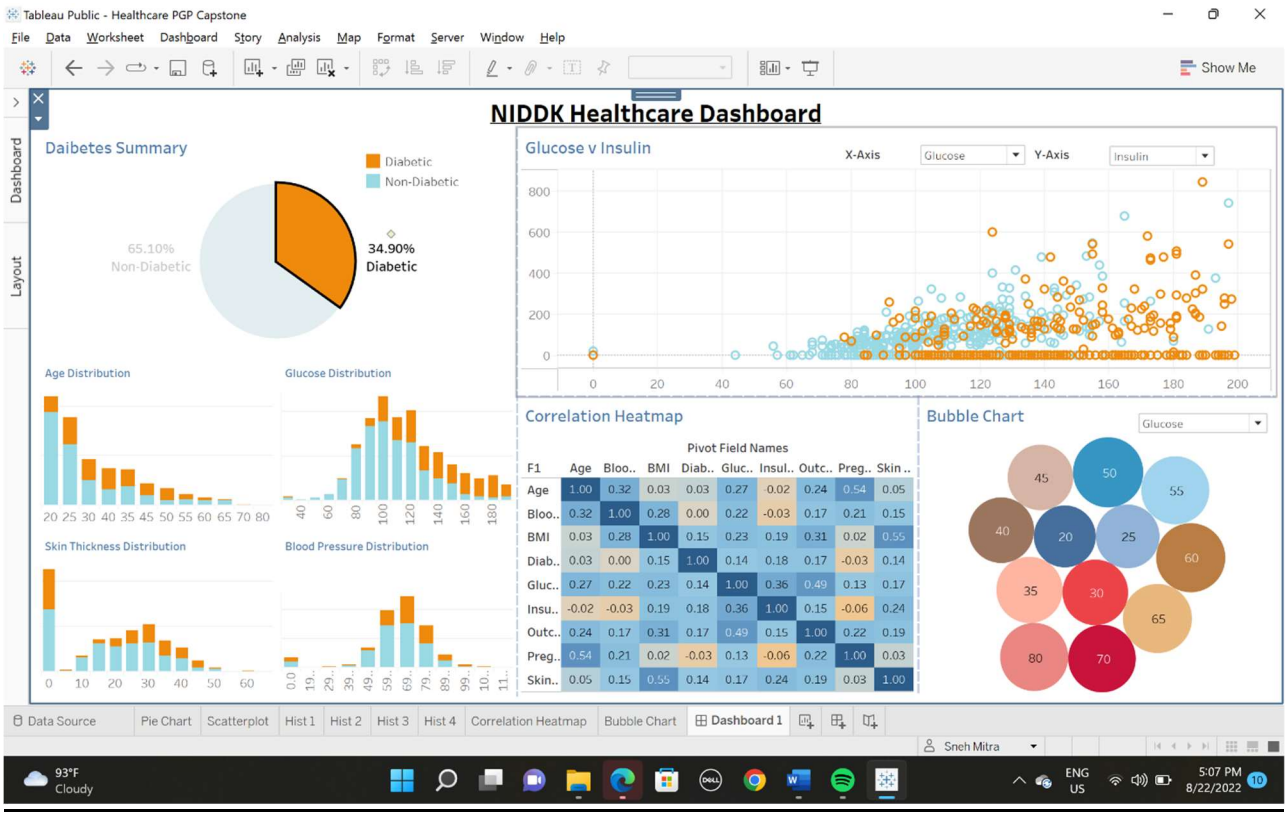


Tableau Public - Healthcare PGP Capstone
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Show Me
NIDDK Healthcare Dashboard
Daibetes Summary
Diabetic
Non-Diabetic
65.10%
Non-Diabetic
34.90%
Diabetic
Glucose v Insulin
X-Axis
Glucose
Y-Axis
Insulin
Age Distribution
Glucose Distribution
Skin Thickness Distribution
Blood Pressure Distribution
Correlation Heatmap
Pivot Field Names
Bubble Chart
Glucose
Data Source
Pie Chart
Scatterplot
Hist 1
Hist 2
Hist 3
Hist 4
Correlation Heatmap
Bubble Chart
Dashboard 1
Sneh Mitra
93°F
Cloudy
ENG
US
5:07 PM
8/22/2022