# CUNY MSDS DATA622 - Machine Learning & Big Data

## Homework #2 : Decision Tree and Classification or Regression problem

### Dataset : National Institute of Diabetes and Digestive and Kidney Diseases

### Ramnivas Singh

---

**Summary**

- Read the following articles:
  - https://www.hindawi.com/journals/complexity/2021/5550344/
  - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8137961/
- Search for academic content (at least 3 articles) that compare the use of decision trees vs SVMs in your current area of expertise.
- Perform an analysis of the dataset used in Homework #2 using the SVM algorithm.
- Compare the results with the results from previous homework.
- Answer questions, such as:
  - Which algorithm is recommended to get more accurate results?
  - Is it better for classification or regression scenarios?
  - Do you agree with the recommendations?
  - Why?

## Search for academic content that compare the use of decision trees vs SVMs in your current area of expertise.

### Credit risk management

Credit risk management has played a key role in financial and banking industry. Inferior credit risk assessment tool is the primary reason of enterprise bankruptcy. Generally speaking, credit risk management includes credit risk analysis, assessment (measurement) of enterprise credit risk and how to manage the risk efficiently, while credit risk assessment is the basic and critical factor in credit risk management.

The main purpose of credit risk assessment is to measure the default possibility of borrowers and provide the loaner a decision-aid by conducting qualitative analysis and qualitative computation to the possible factors that will cause credit risk. At present, the classification is the most popular method used in credit risk assessment.

Predicting the creditworthiness of bank customers is a major concern for banking institutions, as modeling the probability of default is a key focus of the Basel regulations. Practitioners propose different default modeling techniques such as linear discriminant analysis, logistic regression, Bayesian approach, and artificial intelligence techniques. The performance of the default prediction is evaluated by the Receiver Operating Characteristic

(ROC) curve using three types of kernels, namely, the polynomial kernel, the linear kernel and the Gaussian kernel.

SVM is a kind of general forward-feedback network, brought forward by Vapnik first. The idea of SUM is to build up a hyperplane as a decision-making surface which make the isolated margin maximum between positive and negative examples. Moreover, SUM is an approximate method of minimizing structure risks. The idea is based on such fact that error rate of testing data is limited in the sum of training error rate and an item depending on Vapnik Chervonenkis dimension. Under the pattern of SUM classifier, the formal value is zero, the latter value is minimized.

The empirical evidence is in favor of the SVM for classification, especially in the linear non-separable case. The sensitivity investigation and a corresponding visualization tool reveal that the classifying ability of SVM appears to be superior over a wide range of SVM parameters. In terms of the empirical results obtained by SVM, the eight most important predictors related to bankruptcy for these German firms belong to the ratios of activity, profitability, liquidity, leverage and the percentage of incremental inventories. Some of the financial ratios selected by the SVM model are new because they have a strong nonlinear dependence on the default risk but a weak linear dependence that therefore cannot be captured by the usual linear models such as the DA and logit models.

## Support Vector Machines for Prediction of Futures Prices

Stock price prediction is one of the most widely studied and challenging problems, attracting researchers from many fields including economics, history, finance, mathematics, and computer science. The volatile nature of the stock market makes it difficult to apply simple time-series or regression techniques. Financial institutions and traders have created various proprietary models to try and beat the market for themselves or their clients, but rarely has anyone achieved consistently higher-than-average returns on investment.

The purpose of predictive stock price systems is to provide abnormal returns for financial market operators and serve as a basis for risk management tools. Although the Efficient Market Hypothesis (EMH) states that it is not possible to anticipate market movements consistently, the use of computationally intensive systems that employ machine learning algorithms is increasingly common in the development of stock trading mechanisms. Several studies, using daily stock prices, have presented predictive system applications trained on fixed periods without considering new model updates. In this context, study uses a machine learning technique called Support Vector Regression (SVR) to predict stock prices for large and small capitalisations and in three different markets, employing prices with both daily and up-to-the-minute frequencies. Prediction errors are measured, and the model is compared to the random walk model proposed by the EMH. The results suggest that the SVR has predictive power, especially when using a strategy of updating the model periodically. There are also indicative results of increased predictions precision during lower volatility periods.

## Loan Status Prediction using Support Vector Machine

Bank Loan endorsement is a vital cycle for banking associations. This framework endorses or rejects the credit applications. Reimbursement of credit is a significant contributing boundary in the fiscal reports of a bank. It is truly challenging to foresee the chances of reimbursement of credit by the client. Lately numerous analysts chipped away at credit endorsement forecast frameworks. In the System Machine Learning (ML) techniques are extremely helpful in foreseeing results for enormous measure of information. In this paper two AI calculations- Support Vector Machine (SVM) and Random Forest (RF) are applied to anticipate the advance endorsement of clients.

Loan administration is intended to improve economic buoyancy of loan beneficiaries while the lender gets interest at a specific rate based on diverse circumstances like duration of refund, amount, purpose, and credit track records. A loan enters default when the agreed promissory notes are not fulfilled by the loan beneficiary; precisely when no payment or interest is paid in 90 days. However, intentionally officers that should scrutinize loan applications and decline non- promising applications that are likely to be fluent for the firm.

The revelation of hidden patterns and features in data through data mining has allowed for machine learning solutions to fraud detection. Machine Learning employs data mining approaches and other learning algorithms in building models of what is happening behind some data to end up in making predictions or detections. There are supervised and unsupervised learning techniques used in detecting fraudulent acts in various domains. Labelled data can be expensive and difficult to get, but this is what a supervised learning approach uses. Grouping bank credit transactions into “legitimate” and “fraudulent” is a sample of labeling. Training and learning of input variables are channeled towards these target variables.

## EDA Summary

Support Vector Machine or SVM is a supervised and linear Machine Learning algorithm most commonly used for solving classification problems and is also referred to as Support Vector Classification. There is also a subset of SVM called SVR which stands for Support Vector Regression which uses the same principles to solve regression problems. SVM also supports the kernel method also called the kernel SVM which allows us to tackle non-linearity.

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification, implicitly mapping their inputs into high-dimensional feature spaces.

---

I will be using **National Institute of Diabetes and Digestive and Kidney Diseases** This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective is to predict based on diagnostic measurements whether a patient has diabetes.

Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

This dataset is picked up from

- https://www.kaggle.com/datasets/mathchi/diabetes-data-set?select=diabetes.csv
- https://www.kaggle.com/datasets/mustafaali96/weight-height

# Load Packages

In [21]:

```
# Lets setup python environment and load python libraries for this EDA
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import pyplot
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
%matplotlib inline
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
# Import the necessary modules and libraries
import numpy as np
from sklearn import svm
import sklearn.preprocessing
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.model_selection import train_test_split
from sklearn import svm

import warnings
warnings.simplefilter(action='ignore', category=Warning)
```

# Diabetes Analysis & Prediction

## Load Diabetes and Kidney Diseases Data

In [22]:
```
diabetes_df = pd.read_csv("diabetes.csv")
diabetes_df.head()
```

Out[22]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

In [23]:
```
diabetes_df.describe()
```

Out[23]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | |
|---|---|---|---|---|---|---|---|---|
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768 |
| **mean** | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33 |
| **std** | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21 |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 | 24 |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 29 |
| **75%** | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 0.626250 | 41 |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 | 81 |

## Data Exploration

This dataset has following columns, few columns are unused, few are used as an inputs and treated as a feature. Class variable (0 or 1) is expected outcome

**Columns :**

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunction: Diabetes pedigree function
- Age: Age (years)
- Outcome: Class variable (0 or 1)

The dependent variable is if the patient is suffering from diabetes or not. Here the dependent column contains binary variable 1 indicating the person is suffering from diabetes and 0 he is not a patient of diabetes. The data set has independent variables as several physiological parameters of a diabetes patient.

In [24]:
```
#descriptive statistics summary
diabetes_df.head() # printing first few rows of the data
```

Out[24]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

In [25]:
```
diabetes_df.tail()    # to show last few rows of the data
```

Out[25]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | 0.171 | 63 | 0 |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | 0.340 | 27 | 0 |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | 0.245 | 30 | 0 |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | 0.349 | 47 | 1 |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | 0.315 | 23 | 0 |

In [26]:
```
diabetes_df.info() # for a quick view of the data
```

```
<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
```

In [27]:
```
diabetes_df.sample(10)  # display a sample of 10 rows from the data
```

Out[27]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **92** | 7 | 81 | 78 | 40 | 48 | 46.7 | 0.261 | 42 | 0 |
| **146** | 9 | 57 | 80 | 37 | 0 | 32.8 | 0.096 | 41 | 0 |
| **78** | 0 | 131 | 0 | 0 | 0 | 43.2 | 0.270 | 26 | 1 |
| **31** | 3 | 158 | 76 | 36 | 245 | 31.6 | 0.851 | 28 | 1 |
| **11** | 10 | 168 | 74 | 0 | 0 | 38.0 | 0.537 | 34 | 1 |
| **137** | 0 | 93 | 60 | 25 | 92 | 28.7 | 0.532 | 22 | 0 |
| **700** | 2 | 122 | 76 | 27 | 200 | 35.9 | 0.483 | 26 | 0 |
| **256** | 3 | 111 | 56 | 39 | 0 | 30.1 | 0.557 | 30 | 0 |
| **504** | 3 | 96 | 78 | 39 | 0 | 37.3 | 0.238 | 40 | 0 |
| **112** | 1 | 89 | 76 | 34 | 37 | 31.2 | 0.192 | 23 | 0 |

In [28]:
```
diabetes_df.describe()    # printing summary statistics of the data
```

Out[28]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | |
|---|---|---|---|---|---|---|---|---|
| **count** | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768.000000 | 768 |
| **mean** | 3.845052 | 120.894531 | 69.105469 | 20.536458 | 79.799479 | 31.992578 | 0.471876 | 33 |
| **std** | 3.369578 | 31.972618 | 19.355807 | 15.952218 | 115.244002 | 7.884160 | 0.331329 | 11 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078000 | 21 |
| **25%** | 1.000000 | 99.000000 | 62.000000 | 0.000000 | 0.000000 | 27.300000 | 0.243750 | 24 |
| **50%** | 3.000000 | 117.000000 | 72.000000 | 23.000000 | 30.500000 | 32.000000 | 0.372500 | 29 |
| **75%** | 6.000000 | 140.250000 | 80.000000 | 32.000000 | 127.250000 | 36.600000 | 0.626250 | 41 |
| **max** | 17.000000 | 199.000000 | 122.000000 | 99.000000 | 846.000000 | 67.100000 | 2.420000 | 81 |

In [29]:
```
pd.isnull(diabetes_df)  # check for any null values in the data
```

Out[29]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | False |
| **2** | False | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | False | False | False | False | False | False |

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **763** | False | False | False | False | False | False | False | False | False |
| **764** | False | False | False | False | False | False | False | False | False |
| **765** | False | False | False | False | False | False | False | False | False |
| **766** | False | False | False | False | False | False | False | False | False |
| **767** | False | False | False | False | False | False | False | False | False |

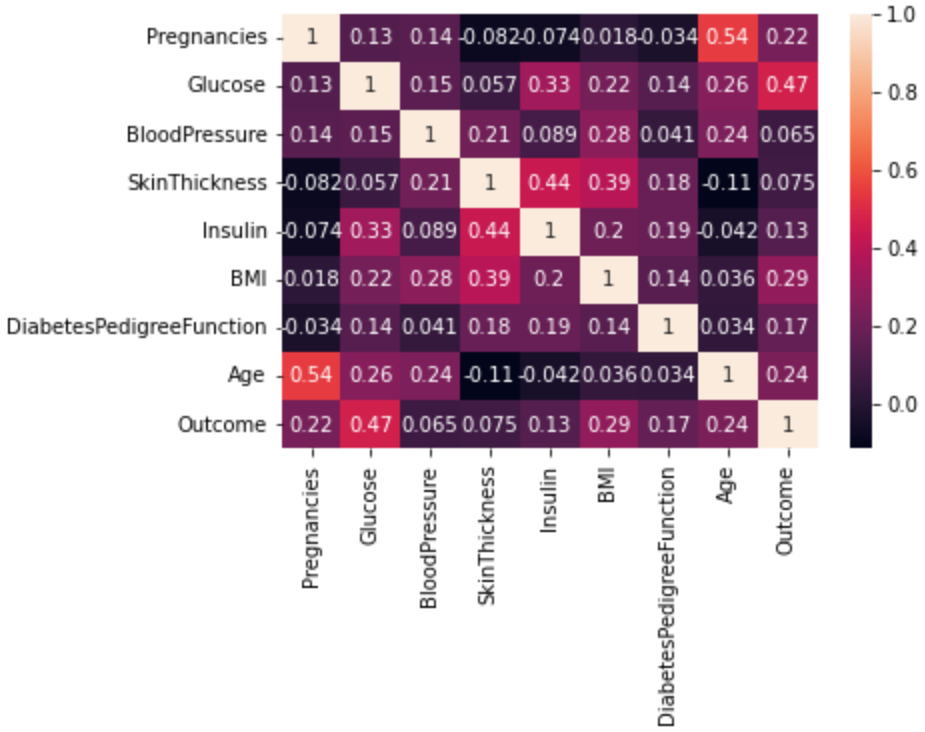
768 rows × 9 columns

The first eight columns contain the independent variables. These are some physiological variables having a correlation with diabetes symptoms. As we can see that the data frame contains nine variables in nine columns. The ninth column shows if the patient is diabetic or not. So, here the x stores the independent variables and y stores the dependent variable diabetes count.

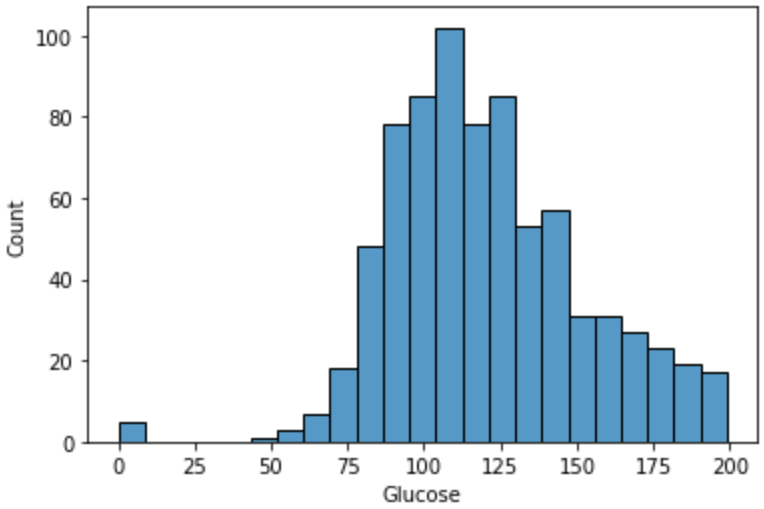
In [30]:
```
x=diabetes_df.iloc[:,:-1].values
y=diabetes_df.iloc[:,-1].values
```

In [31]:
```
corr=diabetes_df.corr()
sns.heatmap(corr, annot=True) # an array of the same shape as data which is used to annota
```

Out[31]:
```
<AxesSubplot:>
```

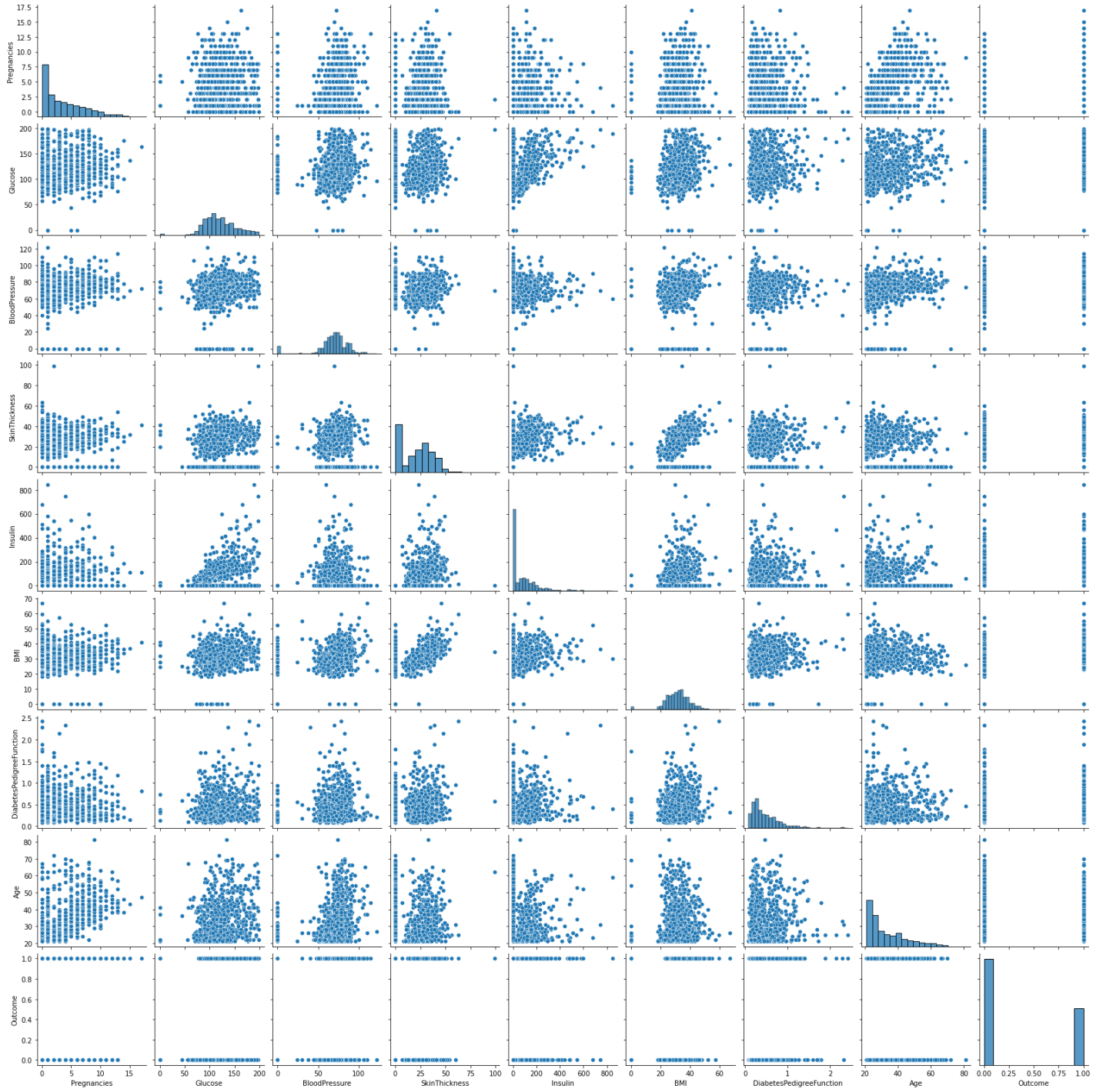



In [32]:
```
ax = sns.histplot(diabetes_df["Glucose"])
plt.show() # Plot histogram to show distributions of dataset
```




In [33]:
```
sns.pairplot(diabetes_df)
```

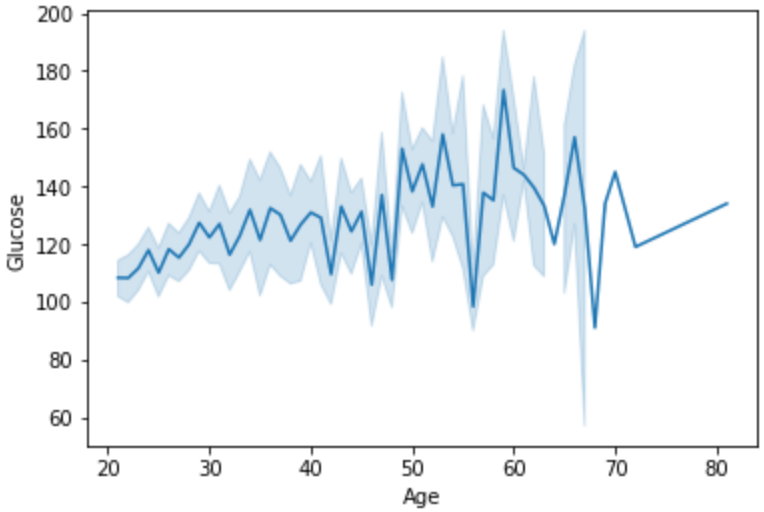
Out[33]:
```
<seaborn.axisgrid.PairGrid at 0x19e7d252160>
```




Fitting the Decision Tree Regression Model to the dataset Create the Decision Tree regressor object. To do the classification we need to import the DecisionTreeClassifier() from sklearn. This special classifier is capable of classifying binary variable i.e. variable with only two classes as well as multiclass variables.

In [34]:
```
sns.lineplot(data=diabetes_df, x="Age", y="Glucose") # plot to see the rent trend
```

Out[34]:
```
<AxesSubplot:xlabel='Age', ylabel='Glucose'>
```




In [35]:
```
zero_not_allowed = ["Glucose","BloodPressure","SkinThickness"]

for column in zero_not_allowed:
    diabetes_df[column] = diabetes_df[column].replace(0, np.NaN)
    mean = int(diabetes_df[column].mean(skipna = True))
    diabetes_df[column] = diabetes_df[column].replace(np.NaN, mean)
```

## SVM Model

In [36]:
```
# Splitting the dataset into training and testing sets.
x = diabetes_df.iloc[:, :-2]
y = diabetes_df.iloc[:, -1]
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 0, test_size = 0.
```

In [37]:
```
# Creating the SVM model.
clf = svm.SVC(kernel='rbf')
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)
```

In [38]:
```
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7922077922077922
```

In [39]:
```
from sklearn.metrics import classification_report, confusion_matrix

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

```
[[98  9]
 [23 24]]
              precision    recall  f1-score   support

           0       0.81      0.92      0.86       107
           1       0.73      0.51      0.60        47

    accuracy                           0.79       154
   macro avg       0.77      0.71      0.73       154
weighted avg       0.78      0.79      0.78       154
```

The Conclusion from Model Building - Therefore Random forest is the best model for this prediction since it has an accuracy_score of 0.79.

# Conclusion

As we can see in the resulting plot, the decision tree of depth 2 captures the general trend in the data. It is important to know how to choose an appropriate value for a depth of a tree to not overfit or underfit the data. Knowing how to combine decision trees to form an ensemble random forest is also useful as it usually has a better generalization performance than an individual decision tree due to randomness, which helps to decrease the model's variance. It is also less sensitive to outliers in the dataset and doesn't require much parameter tuning.

**Home Work 2 (Random Forest Classifier Accuracy: 0.7835497835497836)**

**Home Work 3 (SVM Accuracy: 0.7922077922077922)**

Accuracy from both the models is very close with a thin margin. Considering calculated thin margin - Random Forest Classifier model provides more accuracy and is a better model over SVM for this dataset.

# References


https://scikit-learn.org/stable/modules/tree.html

https://www.tandfonline.com/doi/abs/10.1080/14697680903410015

https://www.sciencedirect.com/science/article/pii/S2405918818300060

https://www.cs.princeton.edu/sites/default/files/uploads/saahil_madge.pdf

http://gide.unileon.es/admin/UploadFolder/journal_of_forecasting.pdf