```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:1
  import pandas.util.testing as tm
```

```
data = pd.read_csv('diabetes.csv')
data.head()
```

	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

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
```

```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insul
count	768.000000	768.000000	768.000000	768.000000	768.0000
mean	3.845052	120.894531	69.105469	20.536458	79.7994
std	3.369578	31.972618	19.355807	15.952218	115.2440
min	0.000000	0.000000	0.000000	0.000000	0.0000
25%	1.000000	99.000000	62.000000	0.000000	0.0000
50%	3.000000	117.000000	72.000000	23.000000	30.5000
75%	6.000000	140.250000	80.000000	32.000000	127.2500
max	17.000000	199.000000	122.000000	99.000000	846.0000

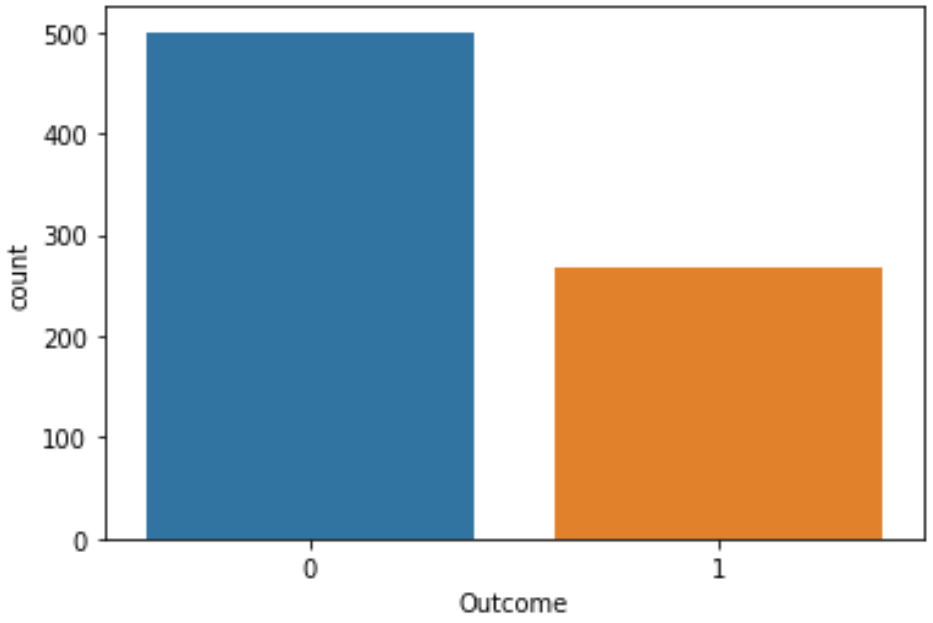
Data Visualizing

```
data.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
import seaborn as sns
sns.countplot(data['Outcome'],label="Count")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66664250b8>
```




```
data.dtypes
```

```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

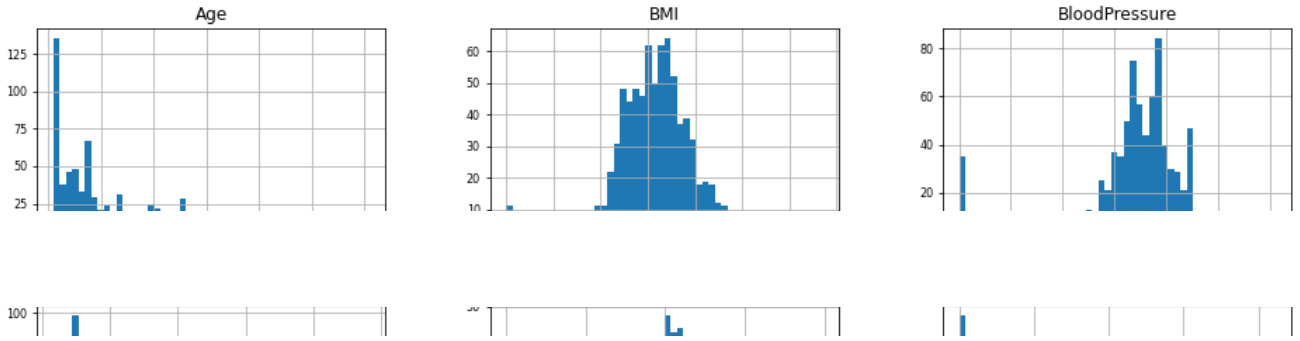
```
data.columns.to_list()
```

```
['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age',
 'Outcome']
```

check outlier and gaussian shape

```
data[['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age',
 'Outcome']].hist(figsize=(16,10),bins=50 ,xlabelsize=8, ylabelsize=8)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f6665ed747
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665f0871
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665ebc97
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6665e6fbe
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665e24e4
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665de801
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f6665d9a35
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665d4e58
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665d4e51
      dtype=object)
```



Outlier Cleaning

Pregnancies more than 10 is ideally not good so we consider it as outlier.

Body mass index is weight to height ration so weight less than 12 is not range of adults so we consider it as outlier.

bloodpressure lower than 40 is criticly low pressure so we consider it as outlier
Glucose lower than 40 is criticly low pressure so we consider it as outlier.

SkinThickness lower than 60 is criticly low pressure so we consider it as outlier

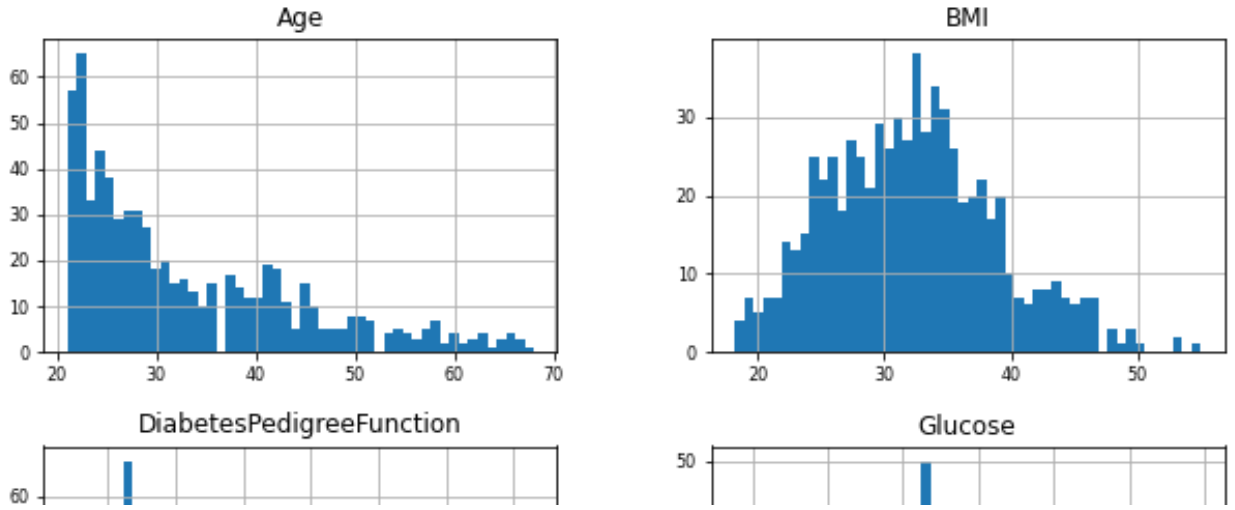
```
from scipy import stats


z_scores = stats.zscore(data)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
new_data = data[filtered_entries]
print(new_data)
```

```
     Pregnancies  Glucose  ...  Age  Outcome
0              6      148  ...   50        1
1              1       85  ...   31        0
2              8      183  ...   32        1
3              1       89  ...   21        0
5              5      116  ...   30        0
..           ...      ...  ...  ...      ...
763           10      101  ...   63        0
764            2      122  ...   27        0
765            5      121  ...   30        0
766            1      126  ...   47        1
767            1       93  ...   23        0
```

```
new_data[['Pregnancies',
 'Glucose',
 'BloodPressure',
 'SkinThickness',
 'Insulin',
 'BMI',
 'DiabetesPedigreeFunction',
 'Age',
 'Outcome']].hist(figsize=(16,10),bins=50 ,xlabelsize=8, ylabelsize=8)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f66663cd82
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f66656eec5
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f666567a5c
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f666561894
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f6665648cc
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f666560608
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f66655b846
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f666556a74
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f666556a7b
      dtype=object)
```

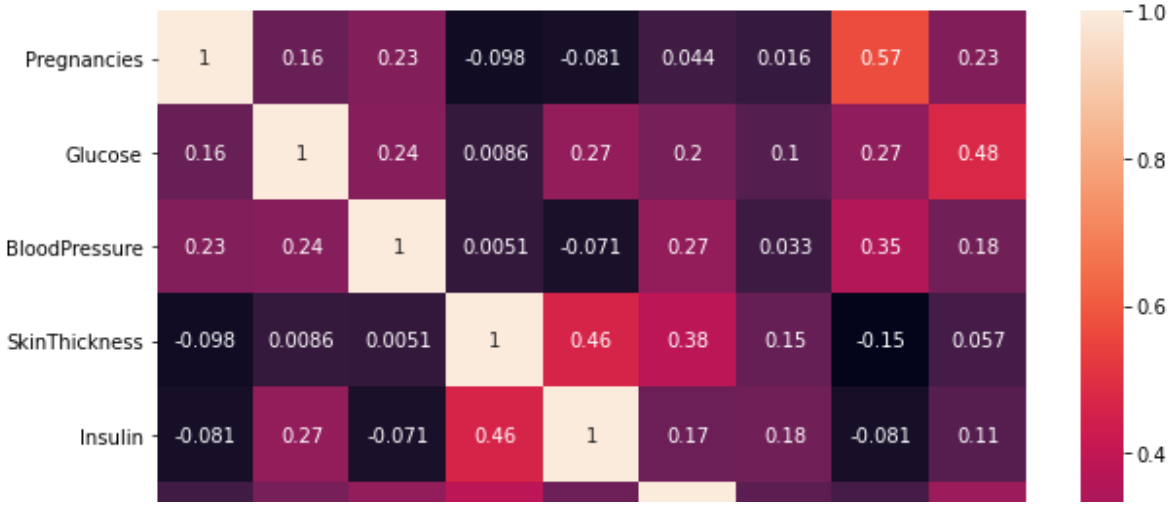



```
sns.pairplot(new_data, hue="Outcome", diag_kind='kde')
```

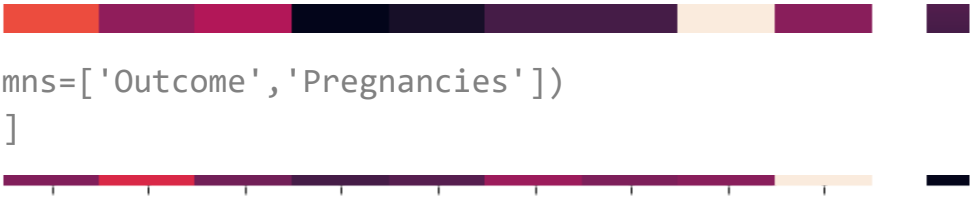
```
<seaborn.axisgrid.PairGrid at 0x7f667619a780>
```

Correlations

```
sns.heatmap(new_data.corr(),annot=True)
fig=plt.gcf()
fig.set_size_inches(10,8)
plt.show()
```



Evaluating for multiple models



```
X=new_data.drop(columns=['Outcome','Pregnancies'])
y=new_data['Outcome']
```

```
X
```

```
Glucose  BloodPressure  SkinThickness  Insulin  BMI  DiabetesPe
```

```
y
```

```
0      1
1      0
2      1
3      0
5      0
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 688, dtype: int64
```

```
#Splitting train test data 80 20 ratio
from sklearn.model_selection import train_test_split
train_X,test_X,train_y,test_y=train_test_split(X,y,test_size=0.2)
```

Applying SVM

```
from sklearn.svm import SVC
svc = SVC( gamma = 'auto')
svc.fit(train_X, train_y)
print("Accuracy on training set: {:.2f}".format(svc.score(train_X, train_y)
print("Accuracy on test set: {:.2f}".format(svc.score(test_X, test_y)))
```

```
Accuracy on training set: 1.00
Accuracy on test set: 0.67
```

The model overfits quite substantially, with a perfect score on the training set and only 71% accuracy on the test set.

SVM requires all the features to vary on a similar scale.

We will need to re-scale our data that all the features are approximately on the same scale:

decision tree classifier

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(random_state=12)
DT.fit(train_X, train_y)

print("Accuracy on training set: {:.2f}".format(DT.score(train_X, train_y))
print("Accuracy on test set: {:.2f}".format(DT.score(test_X, test_y)))
```

```
Accuracy on training set: 1.00
Accuracy on test set: 0.72
```

random forest

```
rf = RandomForestClassifier(n_estimators=420,random_state=72)
model_rf = rf.fit(train_X,train_y)

print("Accuracy on training set: {:.2f}".format(model_rf.score(train_X, tra
print("Accuracy on test set: {:.2f}".format(model_rf.score(test_X, test_y))
```

```
Accuracy on training set: 1.00
Accuracy on test set: 0.75
```

gaussian naive bayes

```
gnb = GaussianNB()
modelgnb = gnb.fit(train_X,train_y)

print("Accuracy on training set: {:.2f}".format(modelgnb.score(train_X, tra
```

```
print("Accuracy on test set: {:.2f}".format(modelgnb.score(test_X, test_y))
```

```
Accuracy on training set: 0.79
Accuracy on test set: 0.70
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(train_X,train_y)

print("Accuracy on training set: {:.2f}".format(modelgnb.score(train_X, tra
print("Accuracy on test set: {:.2f}".format(modelgnb.score(test_X, test_y))
```

```
Accuracy on training set: 0.79
Accuracy on test set: 0.70
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/_logistic.
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as show
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
y_pred = rf.predict(test_X)
```

```
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0])
```

```
test_y
```

```
test_y
```

```
574    0
284    1
361    0
248    0
61     1
      ..
531    0
503    0
396    0
627    0
410    0
Name: Outcome, Length: 138, dtype: int64
```

```
y_pred = rf.predict([[148,72,35,0,33.6,0.627,50]])
y_pred
```

```
array([1])
```

```
y_pred = rf.predict([[131,68,21,166,33.1,0.16,28]])
y_pred
```

```
array([0])
```

Save Model

```
from sklearn.externals import joblib
# Save to file in the current working directory
joblib_file = "Diabetes_Pred.pkl"
joblib.dump(rf, joblib_file)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/externals/joblib/__init
  warnings.warn(msg, category=FutureWarning)
['Diabetes_Pred.pkl']
```

```
# Load from file
model = joblib.load("Diabetes_Pred.pkl")
```

```
model = joblib.load('Diabetes_Pred.pkl')
```

```
result = model.predict([[131,68,21,166,33.1,0.16,28]])
result
```

```
array([0])
```