**CS6220 Data Mining Techniques - Final Project**

Team Members: Paul Cruz, Chih-Ming Sun, Haocheng Yang

GitHub Repo: <https://github.com/Paulrkcruz/CS6220-Final_Project-Diabetes>

**Prediction of Diabetes using Data Mining Techniques**

**Abstract**

The remarkable advances in biotechnology and biomedical sciences have led to a significant production of data, such as high throughput genetic sequencing data and clinical information, generated from large Electronic Health Records (EHRs). To this end, the application of machine learning and data mining methods in the biomedical sciences is presently, more than ever before, vital and indispensable in our efforts to transform all available information into valuable knowledge. Diabetes is defined as a group of metabolic disorders exerting significant pressure on human health across the world. Extensive amounts of research in all aspects of diabetes have led to the generation of enormous amounts of data. The aim of this study was to ascertain if diabetes is either predictable or preventable to the highest accuracy by employing data mining techniques to a publicly available dataset from the University of California, Irvine. Therefore, three machine learning classification models namely Random Forrest, Gradient Boosting, and AdaBoost. The performance of each model was evaluated on various measures like Precision, receiver operating characteristic curve (ROC), and the correlative coefficient. Accuracy is measured over correctly and incorrectly classified instances. By focusing on two features of the data found to be the most significant, Glucose levels and BMI, application and analysis of selected models allowed for proper analysis. The results obtained show that the Random Forrest Classifier outperforms with the highest accuracy of 75% comparatively against the other two models. These results are verified using Receiver Operating Characteristic (ROC) curves in a proper and systematic manner. As a result, the model was shown to be very useful for the prediction of Diabetes.

1. **Introduction**

Diabetes is a well-known disease affecting individuals all over the world. According to the WHO (World Health Association), an estimated 422 million people worldwide currently have diabetes with the majority living in low-and middle-income countries. In addition, 1.6 million deaths are directly attributed to diabetes each year. Both the number of cases and the prevalence of diabetes have been steadily increasing over the past few decades and are expected to grow. But can diabetes be prevented?

Recent research utilizing data science via data mining techniques including predictive modeling using machine learning is inspiring confidence in the medical world (Huang, 2021). Classification strategies are broadly used in the medical field for classifying data into different classes according to individual classifiers. Diabetes, being a disease that affects the human body’s ability to produce the hormone insulin, which in turn causes metabolism of carbohydrates abnormal while at the same time causing elevated levels of glucose in the blood. Generally, an individual with diabetes suffers from high blood sugar, with intensified hunger, frequent urination, being caused by this abnormal level of sugar in the blood. Though there are many factors that may contribute to a diabetes diagnosis, such as heredity, weight, and height, the major factor is blood sugar concentration. The early identification of abnormal glucose levels may be the only remedy to further complications and therefore a more serious diagnosis. Diabetes is a serious health matter during which the measure of sugar substance cannot be controlled. Thus, analysis of the levels of sugar in the blood and how it may compare to other compounds measured is paramount in predictive modeling.

The problem in the diagnosis of diabetes is that many times the diagnosis could have been much sooner, we need a way to diagnose early. In this work we aim to show how this can be done using the machine learning algorithmic approaches of Data mining. We focus on over 70 datasets recorded on patients with diabetes ranging from several weeks to several months, containing readings of glucose, insulin, and patient lifestyle data. Decision Tree machine learning classification algorithms are used and evaluated on the University of Irvine dataset with the aim to predict diabetes in a patient. Therefore, experimental performance of many of the most commonly used predictive modelling algorithms and methods will be employed with the intent to solve this problem.

1. **Methodology**
   1. **Dataset**

The dataset used for this study was provided by Kaggle, a data repository website. This dataset was provided to the repository courtesy of the University of California, Irvine (UCI), and included 70 sets of data recorded on diabetes patients ranging from several weeks to several months’ worth of glucose, insulin, and lifestyle data per patient, as well as a description of the problem domain. The dataset included the following tables: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, Age, and Outcome.

* 1. **Data Processing**

The data was first viewed to allow for proper analysis of its standard distribution. The data was then pre-processed to correct data inconsistencies, missing values, and to remove null values.

* 1. **Data Analysis**

Various types of data mining tools are currently available with each having its own merits and demerits. For this analysis, we first created a heatmap to understand the correlation coefficient of the features of the data. This allowed us to clearly see the correlations between each feature attributes. Next, we chose to plot a histogram to analyze the association between outcome and diagnosis. We then chose to evaluate the following models by splitting the data into 70% training, and 30% test: Decision Tree, Support Vector Machine (SVC), K Neighbors Classifier (KNN), Gradient Boosting Classifier, Random Forrest Classifier, Ada Boost, Gaussian NB, and Gradient Boosting. A cross validation accuracy score was assessed as well. We then trained the machine learning models and evaluated the accuracy of each model. Three models were chosen to move forward with the analysis, in which a confusion matrix and ROC curve was plotted and analyzed. After analyzing feature importance and accuracy scores, two features were found to have significance.

1. **Code**

import numpy as np

import pandas as pd

import itertools

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, make\_scorer, accuracy\_score, roc\_curve, roc\_auc\_score

from sklearn.model\_selection import train\_test\_split, KFold, cross\_val\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.gaussian\_process import GaussianProcessClassifier

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,GradientBoostingClassifier

from sklearn.naive\_bayes import GaussianNB

sns.set()

dataset = pd.read\_csv('./Datasets/diabetes.csv')

dataset.head()

Out[316]:

|  | 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 [317]:

dataset.info(verbose=True)

<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 [318]:

dataset.describe().T

Out[318]:

|  | count | mean | std | min | 25% | 50% | 75% | max |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.0000 | 6.00000 | 17.00 |
| Glucose | 768.0 | 120.894531 | 31.972618 | 0.000 | 99.00000 | 117.0000 | 140.25000 | 199.00 |
| BloodPressure | 768.0 | 69.105469 | 19.355807 | 0.000 | 62.00000 | 72.0000 | 80.00000 | 122.00 |
| SkinThickness | 768.0 | 20.536458 | 15.952218 | 0.000 | 0.00000 | 23.0000 | 32.00000 | 99.00 |
| Insulin | 768.0 | 79.799479 | 115.244002 | 0.000 | 0.00000 | 30.5000 | 127.25000 | 846.00 |
| BMI | 768.0 | 31.992578 | 7.884160 | 0.000 | 27.30000 | 32.0000 | 36.60000 | 67.10 |
| DiabetesPedigreeFunction | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | 0.3725 | 0.62625 | 2.42 |
| Age | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | 29.0000 | 41.00000 | 81.00 |
| Outcome | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | 0.0000 | 1.00000 | 1.00 |

In [319]:

print(dataset.isnull().sum())

Pregnancies 0

Glucose 0

BloodPressure 0

SkinThickness 0

Insulin 0

BMI 0

DiabetesPedigreeFunction 0

Age 0

Outcome 0

dtype: int64

In [320]:

dataset[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = dataset[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)

In [321]:

print(dataset.isnull().sum())

Pregnancies 0

Glucose 5

BloodPressure 35

SkinThickness 227

Insulin 374

BMI 11

DiabetesPedigreeFunction 0

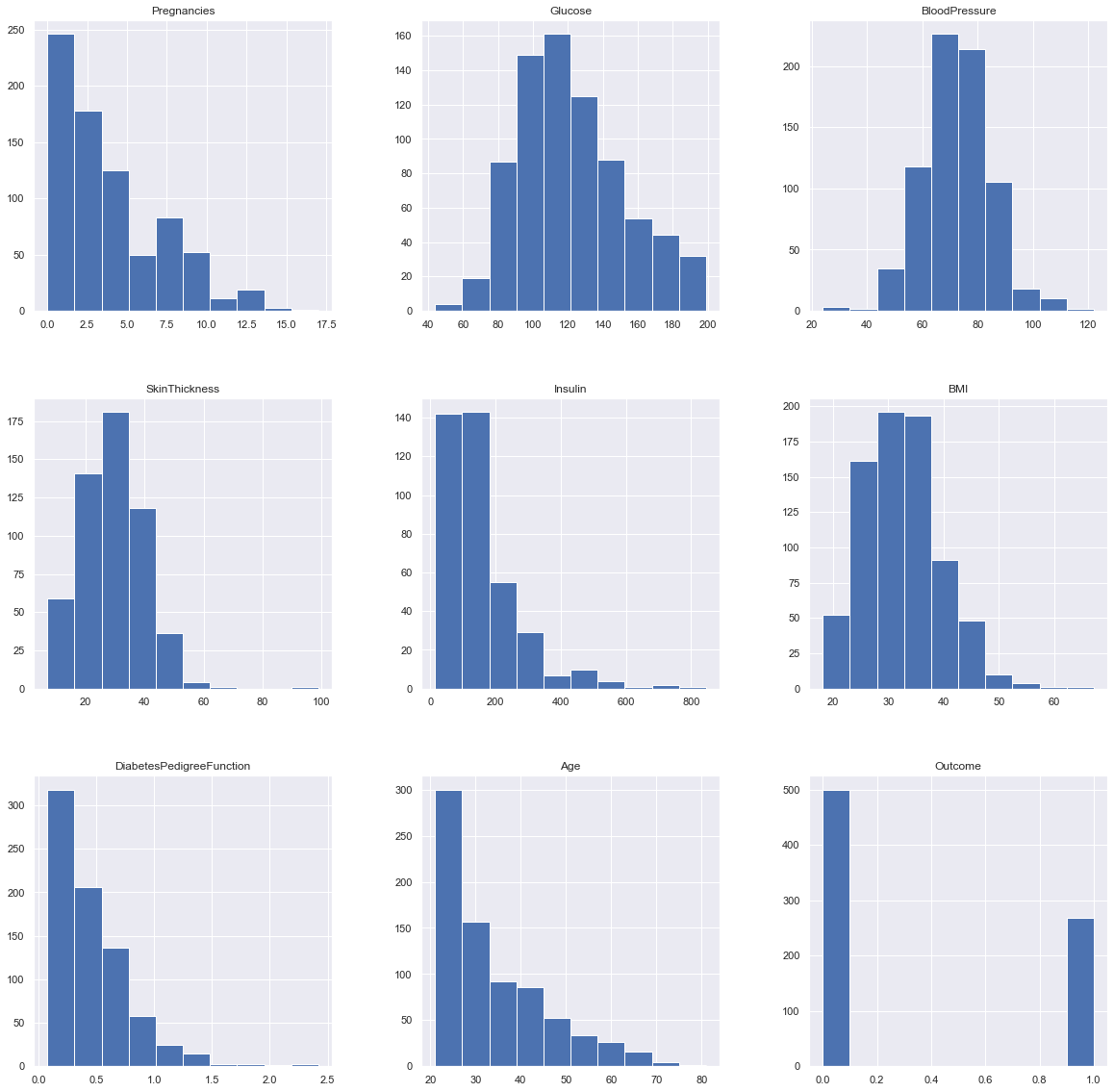
Age 0

Outcome 0

dtype: int64

In [322]:

p = dataset.hist(figsize = (20,20))



In [323]:

# after analyzing the data distribution, we can choose proper methods to replace nan columns to meanningful value.

dataset['Glucose'].fillna(dataset['Glucose'].mean(), inplace = True)

dataset['BloodPressure'].fillna(dataset['BloodPressure'].mean(), inplace = True)

dataset['SkinThickness'].fillna(dataset['SkinThickness'].median(), inplace = True)

dataset['Insulin'].fillna(dataset['Insulin'].median(), inplace = True)

dataset['BMI'].fillna(dataset['BMI'].median(), inplace = True)

In [324]:

dataset.describe().T

Out[324]:

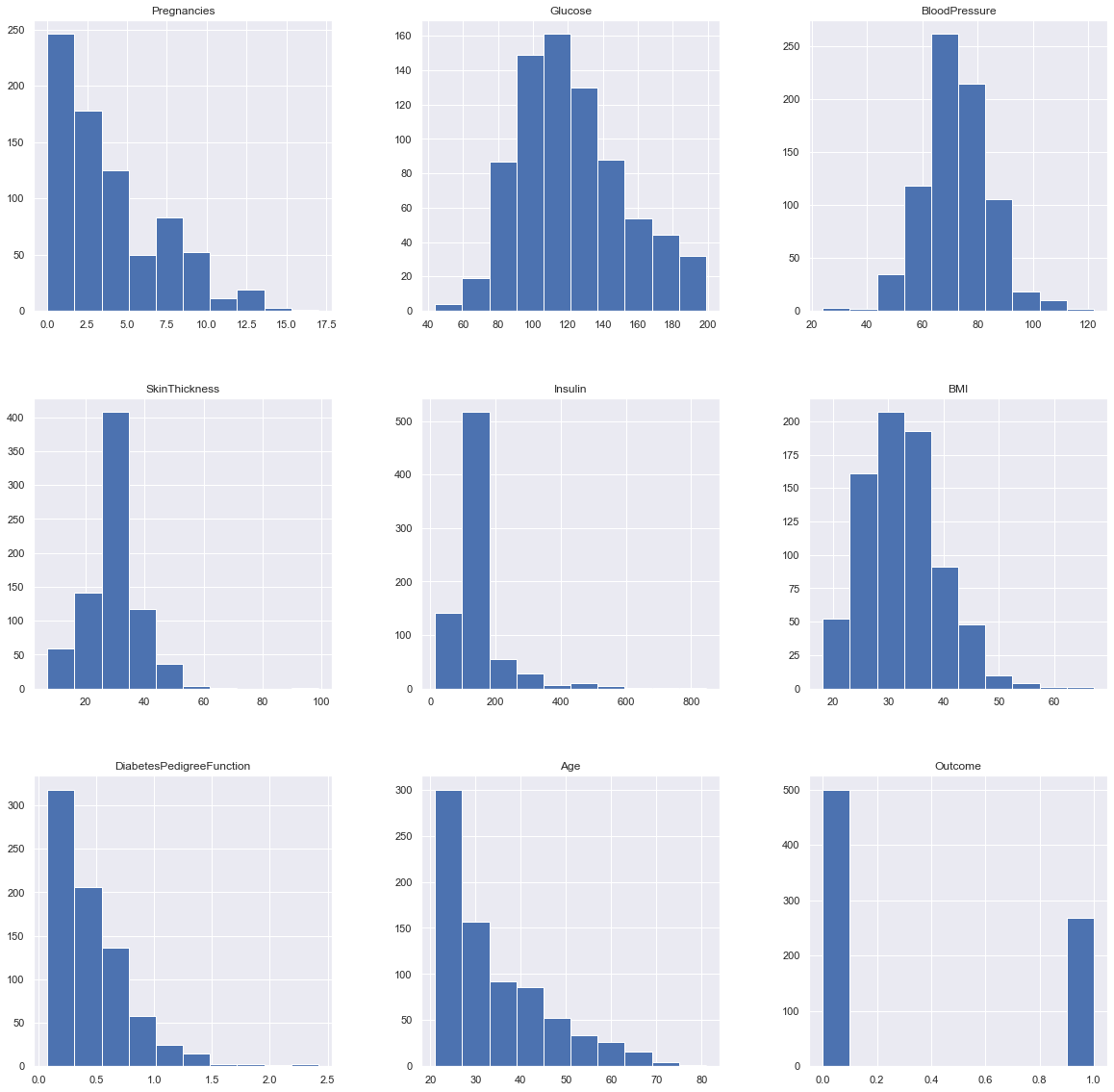
|  | count | mean | std | min | 25% | 50% | 75% | max |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | 768.0 | 3.845052 | 3.369578 | 0.000 | 1.00000 | 3.000000 | 6.00000 | 17.00 |
| Glucose | 768.0 | 121.686763 | 30.435949 | 44.000 | 99.75000 | 117.000000 | 140.25000 | 199.00 |
| BloodPressure | 768.0 | 72.405184 | 12.096346 | 24.000 | 64.00000 | 72.202592 | 80.00000 | 122.00 |
| SkinThickness | 768.0 | 29.108073 | 8.791221 | 7.000 | 25.00000 | 29.000000 | 32.00000 | 99.00 |
| Insulin | 768.0 | 140.671875 | 86.383060 | 14.000 | 121.50000 | 125.000000 | 127.25000 | 846.00 |
| BMI | 768.0 | 32.455208 | 6.875177 | 18.200 | 27.50000 | 32.300000 | 36.60000 | 67.10 |
| DiabetesPedigreeFunction | 768.0 | 0.471876 | 0.331329 | 0.078 | 0.24375 | 0.372500 | 0.62625 | 2.42 |
| Age | 768.0 | 33.240885 | 11.760232 | 21.000 | 24.00000 | 29.000000 | 41.00000 | 81.00 |
| Outcome | 768.0 | 0.348958 | 0.476951 | 0.000 | 0.00000 | 0.000000 | 1.00000 | 1.00 |

In [325]:

# Plot again after all NAN are removed.

In [326]:

p = dataset.hist(figsize = (20,20))



In [327]:

#

positive = dataset.loc[dataset['Outcome'] == 1]

negative = dataset.loc[dataset['Outcome'] == 0]

number\_positive\_each\_age = positive.groupby('Age')['Outcome'].count()

number\_negative\_each\_age = negative.groupby('Age')['Outcome'].count()

result = pd.DataFrame(dict(positive = number\_positive\_each\_age, negative = number\_negative\_each\_age)).reset\_index().fillna(0).set\_index('Age')

result.plot.bar(figsize=[20,20])

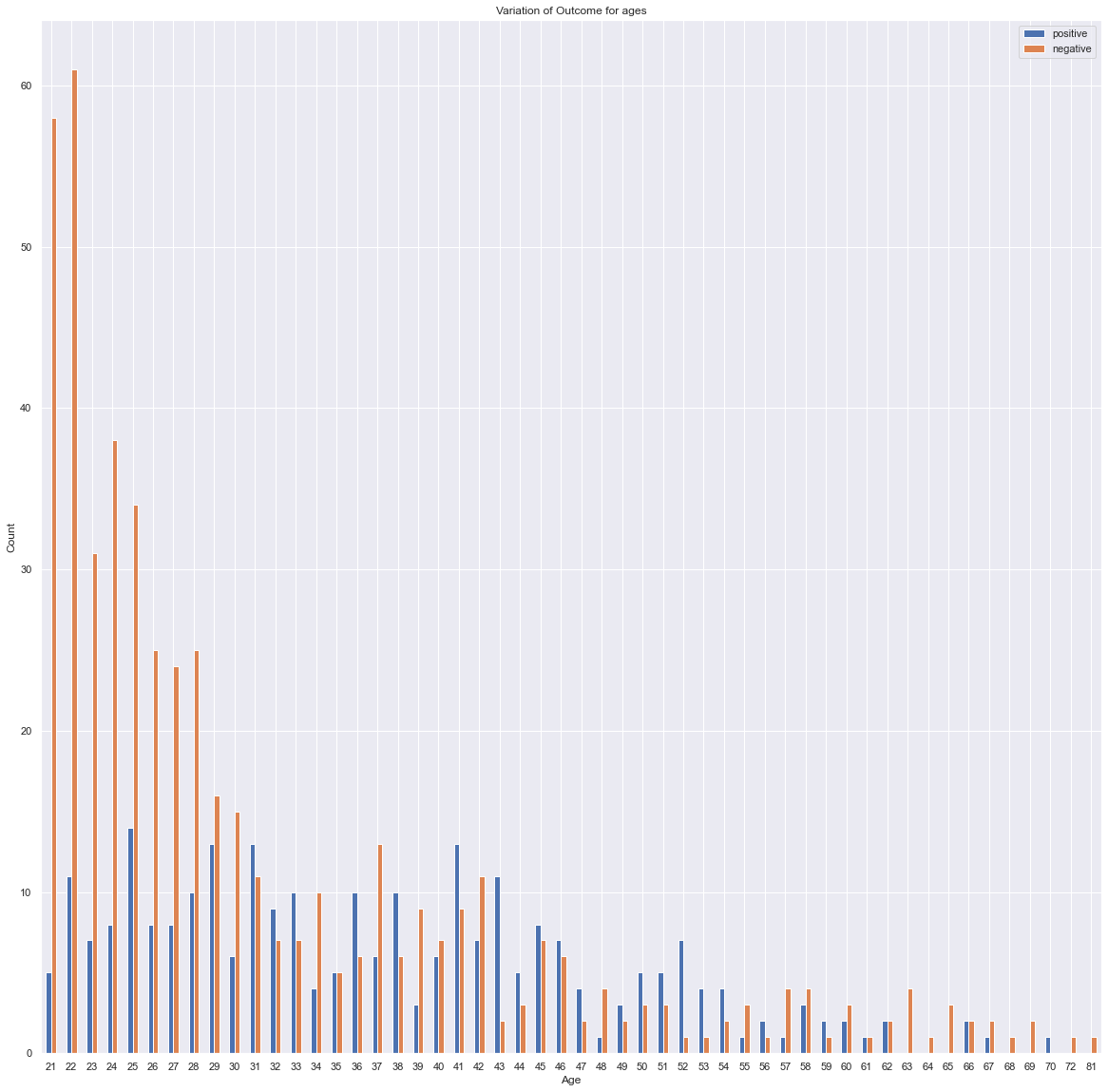
plt.xticks(rotation=360)

plt.title('Variation of Outcome for ages')

plt.ylabel('Count')

plt.xlabel('Age');

plt.show()

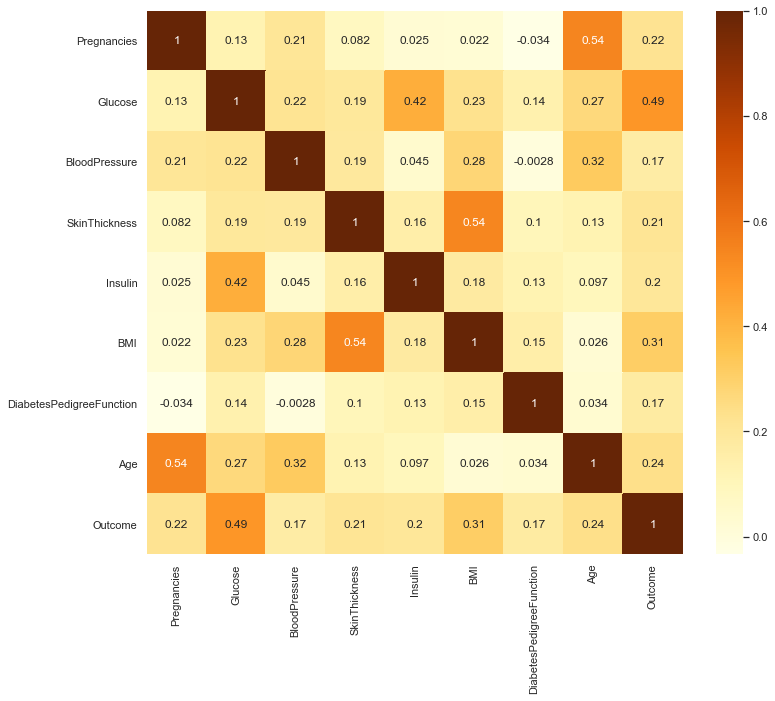


In [328]:

#heatmap

plt.figure(figsize=(12,10)) # on this line I just set the size of figure to 12 by 10.

p=sns.heatmap(dataset.corr(), annot=True,cmap ='YlOrBr')



Analysis:[¶](#Analysis:)

The value of Correlation Coefficient can be between -1 to +1. 1 means that they are highly correlated and 0 means no correlation. According to the heatmap, we can clearly see the correlations between each attributes.[¶](#The-value-of-Correlation-Coefficient-ca)

In [329]:

dataset.corr()

Out[329]:

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Pregnancies | 1.000000 | 0.127911 | 0.208522 | 0.081770 | 0.025047 | 0.021559 | -0.033523 | 0.544341 | 0.221898 |
| Glucose | 0.127911 | 1.000000 | 0.218367 | 0.192686 | 0.419064 | 0.231128 | 0.137060 | 0.266534 | 0.492928 |
| BloodPressure | 0.208522 | 0.218367 | 1.000000 | 0.191853 | 0.045087 | 0.281199 | -0.002763 | 0.324595 | 0.166074 |
| SkinThickness | 0.081770 | 0.192686 | 0.191853 | 1.000000 | 0.155610 | 0.543205 | 0.102188 | 0.126107 | 0.214873 |
| Insulin | 0.025047 | 0.419064 | 0.045087 | 0.155610 | 1.000000 | 0.180241 | 0.126503 | 0.097101 | 0.203790 |
| BMI | 0.021559 | 0.231128 | 0.281199 | 0.543205 | 0.180241 | 1.000000 | 0.153438 | 0.025597 | 0.312038 |
| DiabetesPedigreeFunction | -0.033523 | 0.137060 | -0.002763 | 0.102188 | 0.126503 | 0.153438 | 1.000000 | 0.033561 | 0.173844 |
| Age | 0.544341 | 0.266534 | 0.324595 | 0.126107 | 0.097101 | 0.025597 | 0.033561 | 1.000000 | 0.238356 |
| Outcome | 0.221898 | 0.492928 | 0.166074 | 0.214873 | 0.203790 | 0.312038 | 0.173844 | 0.238356 | 1.000000 |

In [330]:

def plotHistogram(values,label,feature,title):

sns.set\_style("whitegrid")

plotOne = sns.FacetGrid(values, hue=label,aspect=2)

plotOne.map(sns.histplot,feature,kde=False)

plotOne.set(xlim=(0, values[feature].max()))

plotOne.add\_legend()

plotOne.set\_axis\_labels(feature, 'Proportion')

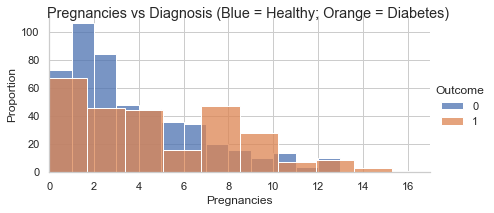
plotOne.fig.suptitle(title)

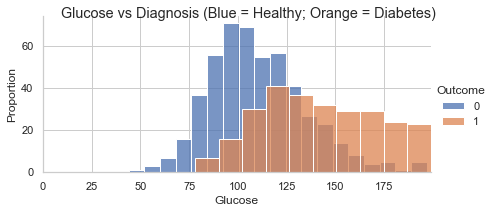
plt.show()

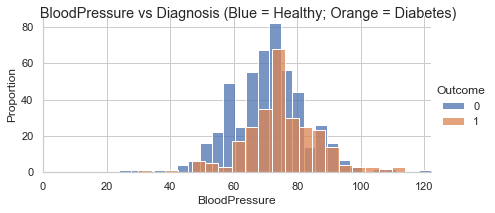
attributes = dataset.columns.values

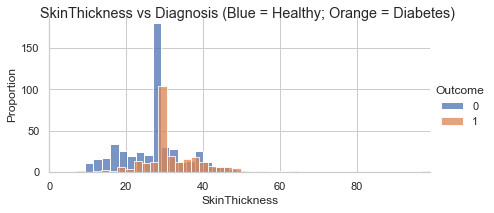
for i in attributes[: -1]:

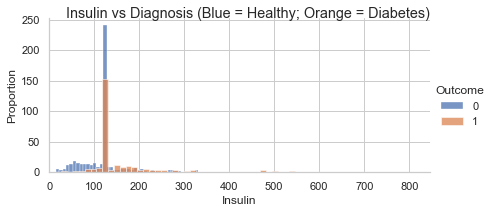
plotHistogram(dataset,"Outcome",i, i + ' vs Diagnosis (Blue = Healthy; Orange = Diabetes)')

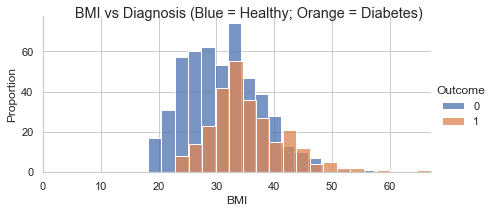


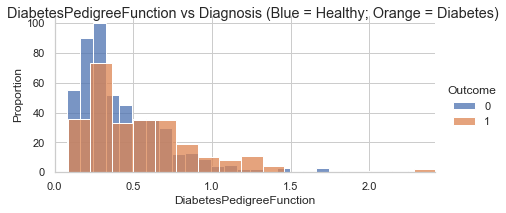


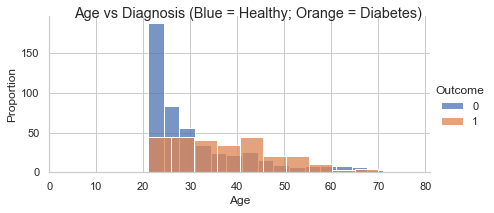












In [1]:

def show\_used\_models():

print('Decision Tree = DecisionTreeClassifier')

print('SVC = Support Vector Machine SVC')

print('KNN = KNeighborsClassifier')

print('GradientBoosting = GradientBoostingClassifier')

print('Gaussian NB = GaussianNB')

print('Random Forest = RandomForestClassifier')

print('Ada Boost = AdaBoostClassifier')

print('GradientBoosting = GradientBoostingClassifier \n\n')

In [332]:

# modelling

X = dataset.iloc[:, :-1]

y = dataset.iloc[:, -1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3)

models = []

model\_names = []

results = []

models.append(('Decision Tree', DecisionTreeClassifier()))

models.append(('SVC', SVC()))

models.append(('KNN', KNeighborsClassifier()))

models.append(('GradientBoosting', GradientBoostingClassifier()))

models.append(('Gaussian NB', GaussianNB()))

models.append(('Random Forest', RandomForestClassifier()))

models.append(('Ada Boost', AdaBoostClassifier()))

for name, model in models:

model.fit(X\_train, y\_train)

kfold = KFold(n\_splits=10)

accuracy\_results = cross\_val\_score(model, X\_train,y\_train, cv=kfold, scoring='accuracy')

results.append(accuracy\_results)

model\_names.append(name)

accuracyMessage = "%s: %f (%f)" % (name, accuracy\_results.mean(), accuracy\_results.std())

print(accuracyMessage)

Decision Tree: 0.677918 (0.051998)

SVC: 0.739693 (0.073575)

KNN: 0.707827 (0.070016)

GradientBoosting: 0.750559 (0.061796)

Gaussian NB: 0.769357 (0.055614)

Random Forest: 0.743291 (0.056324)

Ada Boost: 0.720825 (0.072511)

In [333]:

# boxplot for each model

fig = plt.figure(figsize=[20,20])

fig.suptitle('Algorithm Comparison: Accuracy')

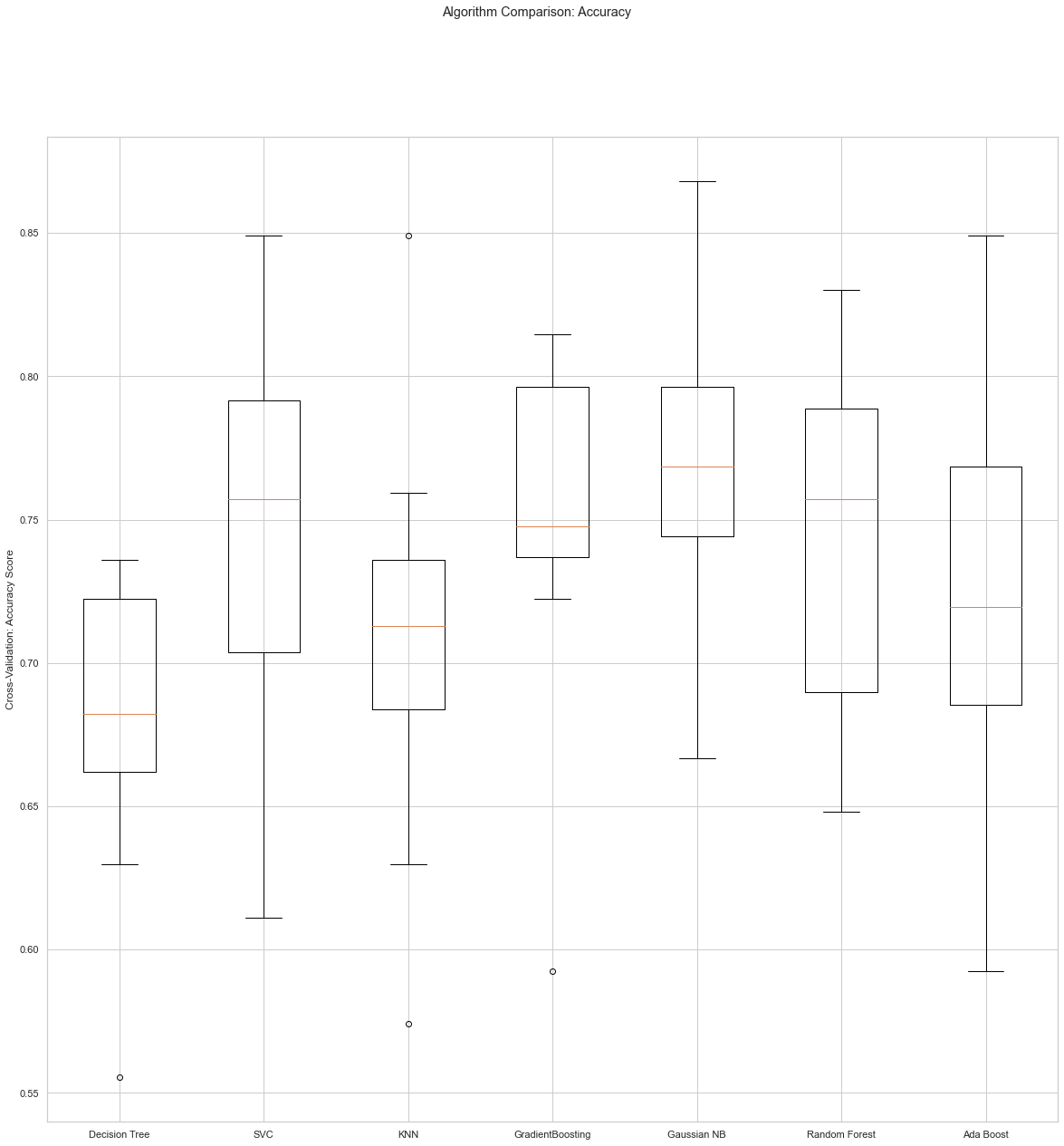
ax = fig.add\_subplot(111)

plt.boxplot(results)

ax.set\_xticklabels(model\_names)

ax.set\_ylabel('Cross-Validation: Accuracy Score')

plt.show()



In [334]:

show\_used\_models()

results = {}

for name, clf in models:

scores = cross\_val\_score(clf, X\_train, y\_train, cv=5)

results[name] = scores

for name, scores in results.items():

print("%20s | Accuracy: %0.2f%% (+/- %0.2f%%)" % (name, 100\*scores.mean(), 100\*scores.std() \* 2))

Decision Tree = DecisionTreeClassifier

SVC = Support Vector Machine SVC

KNN = KNeighborsClassifier

GradientBoosting = GradientBoostingClassifier

Gaussian NB = GaussianNB

Random Forest = RandomForestClassifier

Ada Boost = AdaBoostClassifier

GradientBoosting = GradientBoostingClassifier

Decision Tree | Accuracy: 68.53% (+/- 5.50%)

SVC | Accuracy: 74.50% (+/- 8.48%)

KNN | Accuracy: 72.07% (+/- 8.25%)

GradientBoosting | Accuracy: 73.74% (+/- 9.99%)

Gaussian NB | Accuracy: 76.36% (+/- 4.05%)

Random Forest | Accuracy: 73.18% (+/- 7.40%)

Ada Boost | Accuracy: 70.96% (+/- 9.97%)

In [335]:

def plot\_confusion\_matrix(cm, classes,

normalize=False,

title='Confusion matrix',

cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt),

horizontalalignment="center",

color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

In [336]:

# according to the analysis above, we picked the top 3 high accuracy model to explore more

candidate\_models = [('random forest', RandomForestClassifier()), ('GradientBoosting',GradientBoostingClassifier()), ('AdaBoost',AdaBoostClassifier())]

for name, model in candidate\_models:

model.fit(X\_train, y\_train)

kfold = KFold(n\_splits=10)

accuracy = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')

mean = accuracy.mean()

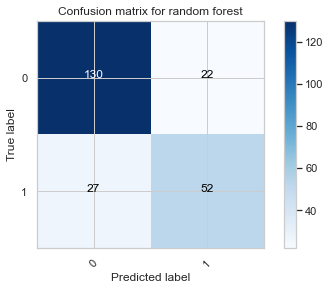
stdev = accuracy.std()

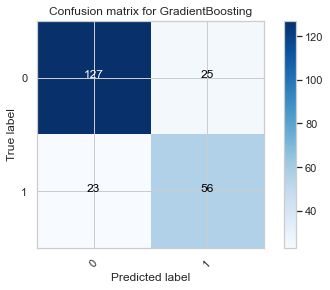
prediction = model.predict(X\_test)

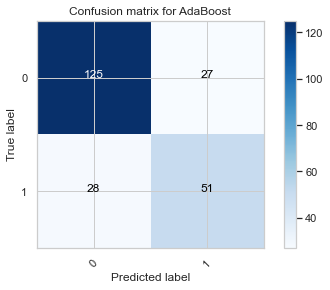
cnf\_matrix = confusion\_matrix(y\_test, prediction)

plot\_confusion\_matrix(cnf\_matrix, classes={0:'Healthy', 1:'Diabetes'},title='Confusion matrix for '+ name)

plt.show()







In [337]:

# ROC

def no\_skill\_prediction():

ns\_probs = [0 for \_ in range(len(y\_test))]

ns\_auc = roc\_auc\_score(y\_test, ns\_probs)

print('No Skill: ROC AUC=%.3f' % (ns\_auc))

ns\_fpr, ns\_tpr, \_ = roc\_curve(y\_test, ns\_probs)

plt.plot(ns\_fpr, ns\_tpr, linestyle='--', label='No Skill')

def graph\_roc\_auc(model, name):

# predict probabilities

lr\_probs = model.predict\_proba(X\_test)

lr\_probs = lr\_probs[:, 1]

# calculate score

lr\_auc = roc\_auc\_score(y\_test, lr\_probs)

# print score

print(name + ': ROC AUC=%.3f' % (lr\_auc))

# calculate roc curves

lr\_fpr, lr\_tpr, \_ = roc\_curve(y\_test, lr\_probs)

# plot the roc curve for the model

plt.plot(lr\_fpr, lr\_tpr, marker='.', label=name)

# axis labels

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

# show the legend

plt.legend()

In [338]:

no\_skill\_prediction()

for name, model in (candidate\_models):

graph\_roc\_auc(model, name)

plt.title('ROC AUC comparison')

No Skill: ROC AUC=0.500

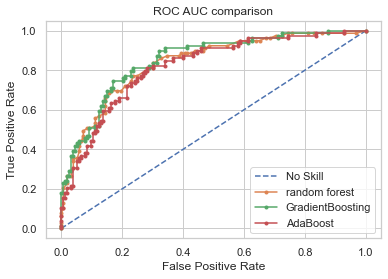
random forest: ROC AUC=0.837

GradientBoosting: ROC AUC=0.851

AdaBoost: ROC AUC=0.818

Out[338]:

Text(0.5, 1.0, 'ROC AUC comparison')



In [339]:

for name, model in candidate\_models:

model.fit(X\_train, y\_train)

print('Accuracy of ' + name + ': {:.2f}'.format(model.score(X\_test, y\_test)))

columns = dataset.columns.values

coefficients = model.feature\_importances\_

absCoefficients = abs(coefficients)

fullList = pd.concat((pd.DataFrame(columns, columns = ['Variable']), pd.DataFrame(absCoefficients, columns = ['absCoefficient'])), axis = 1).sort\_values(by='absCoefficient', ascending = False)

print('DecisionTreeClassifier - Feature Importance:')

print('\n',fullList,'\n')

Accuracy of random forest: 0.79

DecisionTreeClassifier - Feature Importance:

Variable absCoefficient

1 Glucose 0.261964

5 BMI 0.160252

7 Age 0.129872

6 DiabetesPedigreeFunction 0.122828

0 Pregnancies 0.087502

4 Insulin 0.085999

2 BloodPressure 0.083820

3 SkinThickness 0.067764

8 Outcome NaN

Accuracy of GradientBoosting: 0.79

DecisionTreeClassifier - Feature Importance:

Variable absCoefficient

1 Glucose 0.388823

5 BMI 0.202975

7 Age 0.103672

6 DiabetesPedigreeFunction 0.099697

0 Pregnancies 0.073492

4 Insulin 0.073369

2 BloodPressure 0.043211

3 SkinThickness 0.014761

8 Outcome NaN

Accuracy of AdaBoost: 0.76

DecisionTreeClassifier - Feature Importance:

Variable absCoefficient

5 BMI 0.24

1 Glucose 0.18

2 BloodPressure 0.14

6 DiabetesPedigreeFunction 0.14

4 Insulin 0.10

7 Age 0.10

3 SkinThickness 0.06

0 Pregnancies 0.04

8 Outcome NaN

After analyzing the three models, we are able to predict diabetes from medical records with an accuracy of approximately 76%. This was done by focusing on important features such as Glucose levels and BMI.

In [340]:

dataset

Out[340]:

|  | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age | Outcome |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 6 | 148.0 | 72.0 | 35.0 | 125.0 | 33.6 | 0.627 | 50 | 1 |
| 1 | 1 | 85.0 | 66.0 | 29.0 | 125.0 | 26.6 | 0.351 | 31 | 0 |
| 2 | 8 | 183.0 | 64.0 | 29.0 | 125.0 | 23.3 | 0.672 | 32 | 1 |
| 3 | 1 | 89.0 | 66.0 | 23.0 | 94.0 | 28.1 | 0.167 | 21 | 0 |
| 4 | 0 | 137.0 | 40.0 | 35.0 | 168.0 | 43.1 | 2.288 | 33 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 763 | 10 | 101.0 | 76.0 | 48.0 | 180.0 | 32.9 | 0.171 | 63 | 0 |
| 764 | 2 | 122.0 | 70.0 | 27.0 | 125.0 | 36.8 | 0.340 | 27 | 0 |
| 765 | 5 | 121.0 | 72.0 | 23.0 | 112.0 | 26.2 | 0.245 | 30 | 0 |
| 766 | 1 | 126.0 | 60.0 | 29.0 | 125.0 | 30.1 | 0.349 | 47 | 1 |
| 767 | 1 | 93.0 | 70.0 | 31.0 | 125.0 | 30.4 | 0.315 | 23 | 0 |

768 rows × 9 columns

In [341]:

columns = ['Pregnancies', 'BloodPressure', 'DiabetesPedigreeFunction','Age', 'SkinThickness', 'Insulin']

dataset = dataset.drop(columns, axis=1)

In [342]:

dataset

Out[342]:

|  | Glucose | BMI | Outcome |
| --- | --- | --- | --- |
| 0 | 148.0 | 33.6 | 1 |
| 1 | 85.0 | 26.6 | 0 |
| 2 | 183.0 | 23.3 | 1 |
| 3 | 89.0 | 28.1 | 0 |
| 4 | 137.0 | 43.1 | 1 |
| ... | ... | ... | ... |
| 763 | 101.0 | 32.9 | 0 |
| 764 | 122.0 | 36.8 | 0 |
| 765 | 121.0 | 26.2 | 0 |
| 766 | 126.0 | 30.1 | 1 |
| 767 | 93.0 | 30.4 | 0 |

768 rows × 3 columns

In [362]:

X = dataset.iloc[:, :-1]

y = dataset.iloc[:, -1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=1)

clf = GradientBoostingClassifier()

clf.fit(X\_train, y\_train)

print('Accuracy of GradientBoostingClassifier in Reduced Feature Space: {:.2f}'.format(clf.score(X\_test, y\_test)))

columns = dataset.columns

coefficients = clf.feature\_importances\_

absCoefficients = abs(coefficients)

fullList = pd.concat((pd.DataFrame(columns, columns = ['Variable']), pd.DataFrame(absCoefficients, columns = ['absCoefficient'])), axis = 1).sort\_values(by='absCoefficient', ascending = False)

print('\n GradientBoostingClassifier - Feature Importance:')

print('\n',fullList,'\n')

Accuracy of GradientBoostingClassifier in Reduced Feature Space: 0.75

GradientBoostingClassifier - Feature Importance:

Variable absCoefficient

0 Glucose 0.622474

1 BMI 0.377526

2 Outcome NaN

1. **Results**

First, we created a heatmap to understand the correlation coefficient of the features of the data. This allowed us to clearly see the correlations between each feature attributes. Here we analyzed the value of the correlation coefficient, which can be between -1 to +1, respectively. A score of 1 indicates that the features are highly correlated, while a score of 0 means there is no correlation. The heatmap clearly showed correlation between each feature attribute. Next, we chose to plot a histogram to analyze the association between outcome and diagnosis. Preganancy vs diagnosis showed that a around half of pregnancys resulted in an outcome of a women getting diabetes. Next, glucose vs diagnosis showed that glucose levels over 76 and mainly above 125 was correlated with diabetes diagnosis. Blood pressure vs diagnosis showed that both healthy and diabetic individuals have around the same blood pressure, no significance. Insulin vs diagnosis found the same, that health and diabetic patients exhibited similar insulin levels. Diabetes Pedigree Function vs diagnosis showed that around half the proportion exhibited diabetes. And last, age vs diagnosis showed that diabetes in this dataset is found between he gaes of 22-70 years old.

Next, we evaluated the following models by splitting the data into 70% training, and 30% test: Decision Tree, Support Vector Machine (SVC), K Neighbors Classifier (KNN), Gradient Boosting Classifier, Random Forrest Classifier, Ada Boost, Gaussian NB, and Gradient Boosting. We then trained the machine learning models and evaluated the accuracy of each model. We conducted a cross-validation-accuracy score boxplot to assess each model. The Decision Tree indicated accuracy of 68.3%, SVC 74.5%, KNN 72%, Gradient Boosting 73.7%, Gaussian NB 76.4%, Random Forrest 73.2%, and Ada Boost 71% accuracy levels. From the accuracy results, we chose thetop 3 models to explore their capability of predicting diabetes. A confusion matrix was plotted and analyzed. The confusion matrix showed that the Gradient Boosting Classifier would be the best choice for us to proceed with our analysis.

Finally, an ROC AUC comparison was plotted with no skill (baseline), Random Forrest, Gradient Boosting, and Ada Boost. The result of the plot shows that all three models perform very similiarly, butm the Gradient Boosting would be the best predictive model. Two features were found to have the most importance; Glucose and BMI. We plotted the accuracy of the Gradient Boosting Classifier in reduced feature space to be 75%, with Glucose absolute coefficient being 0.62, and BMI 0.37.

1. **Discussion**

The model evaluated in this study shows how data mining techniques via machine learning predictive modelling an be used to predict human disease. Our results show how machine learning and classification algorithms can be used to predict diabetes. The two features that were found to be most important were BMI and glucose. For BMI, there is a certain relationship between the satisfaction rate of blood glucose control and overweight or obesity, which explains the importance of BMI in the classification of control satisfaction. This tells us that lifestyle choices can make a difference between whether you develop diabetes or not.

The results show that there are many predictive models that can be applied, but only certain models will work best which is very dependent on the selected dataset. Therefore, when focusing on a certain disease, several appropriate classification algorithms should be selected based on the characteristics of the dataset. By comparing the classification accuracy of many classification algorithms on the dataset, the most effective classification algorithm can be selected and used as the diagnostic model. In general, the performance of machine learning algorithms is evaluated using predictive accuracy. However, this is not appropriate when the data is imbalanced or the costs of different errors vary remarkedly. In addition, despite the claims that machine learning classification algorithms can generate sufficient and effective decision-making, very few have really permeated the clinical practice. Therefore, the practice of using machine learning algorithms to predict disease should be under further study in the biomedical research and development field, and we hope that this study provided a good example of how this can be done.

1. **Future Work**

One of the most important real-world medical problems is the detection of diabetes at its early stage. In this study, systematic efforts are made in finding a system which results in the prediction of diabetes through data science via predictive modelling algorithms. Though this work may not be the final solution to the prediction of diabetes, it serves as an example of the power data science may have on the prediction of other diseases as well. During this work, three machine learning classification algorithms are chosen and evaluated on various measures. Experimental modelling is then performed on the University of California, Irvine, diabetes dataset. The results determine the adequacy of the system with an achieved accuracy of 79.00% using the Random Forrest classification algorithm. In the future, this system with the use of machine learning classification algorithms may be able to be used to predict or diagnose other diseases as well. The work therefor be extended and improved for the automation of diabetes analysis using the methods described here as well as other machine learning algorithms.

**References**

Huang, R. (2021). Prediction of Pima Indians Diabetes with Machine Learning Algorithms. *UCLA*. ProQuest ID: Huang\_ucla\_0031N\_19508. Merritt ID: ark:/13030/m5md4q60. Retrieved from https://escholarship.org/uc/item/6rh07945

Altıntaş, Ergin. “UCI Diabetes Data Set.” *Kaggle*, 1 May 2020, www.kaggle.com/ealtintas/uci-machine-learning-repository-diabetes-data-set.

World Health Organization (WHO). “Diabetes.” *Kaggle*, 2 April 2021, <https://www.who.int/health-topics/diabetes#tab=tab_1>.