Objective:

To fit different ML models that can predict whether a person is diabetic or not.

Introduction:

The dataset used was the Pima Indian Diabetes dataset from Machine Learning Repository (originally from National Institute of Diabetes and Digestive and Kidney Disease) which contains 8 medical diagnostic attributes and one target variable (i.e, Outcome) of 768 female patients with 34.9% having diabetes (268 patients) which has been taken from Kaggle.

Data Description:

There are 768 rows and 9 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)

Methodology:

1. Data is cleaned and Exploratory Data Analysis is performed in it.
2. Model is fitted with logistic Regression, Decision Tree Classifier and Random Forest Classifier separately.
3. F1 Score, Accuracy Score, Recall Score and Precision Score are estimated of each model and the best model has been identified.
4. The analysis has been performed in python.

Data Cleaning:

```
df.head()
```

	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

```
df.isnull().sum()
```

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

No missing values are present. However from the above functions we can see that Glucose, BloodPressure, SkinThickness, Insulin and BMI have min values of 0 which are not real clinical values. Lets look at counts. A value of 0 for Pregnancies is a real value.

```
print("Number of 0's for Glucose:", df['Glucose'].isin([0]).sum())
print("Number of 0's for Blood Pressure:", df['BloodPressure'].isin([0]).sum())
print("Number of 0's for Skin Thickness:", df['SkinThickness'].isin([0]).sum())
print("Number of 0's for Insulin:", df['Insulin'].isin([0]).sum())
print("Number of 0's for BMI:", df['BMI'].isin([0]).sum())
```

```
Number of 0's for Glucose: 5
Number of 0's for Blood Pressure: 35
Number of 0's for Skin Thickness: 227
Number of 0's for Insulin: 374
Number of 0's for BMI: 11
```

Replacing 0 values in these columns with mean.

```
diabetes_clean = df.copy()
```

```
diabetes_clean['Glucose'] = diabetes_clean['Glucose'].replace(0,df['Glucose'].mean())
diabetes_clean['BloodPressure'] = diabetes_clean['BloodPressure'].replace(0,df['BloodPressure']
diabetes_clean['SkinThickness'] = diabetes_clean['SkinThickness'].replace(0,df['SkinThickness']
diabetes_clean['Insulin'] = diabetes_clean['Insulin'].replace(0,df['Insulin'].mean())
diabetes_clean['BMI'] = diabetes_clean['BMI'].replace(0,df['BMI'].mean())
```

```
diabetes_clean.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcon
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50	
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32	
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21	
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33	

Exploratory Data Analysis

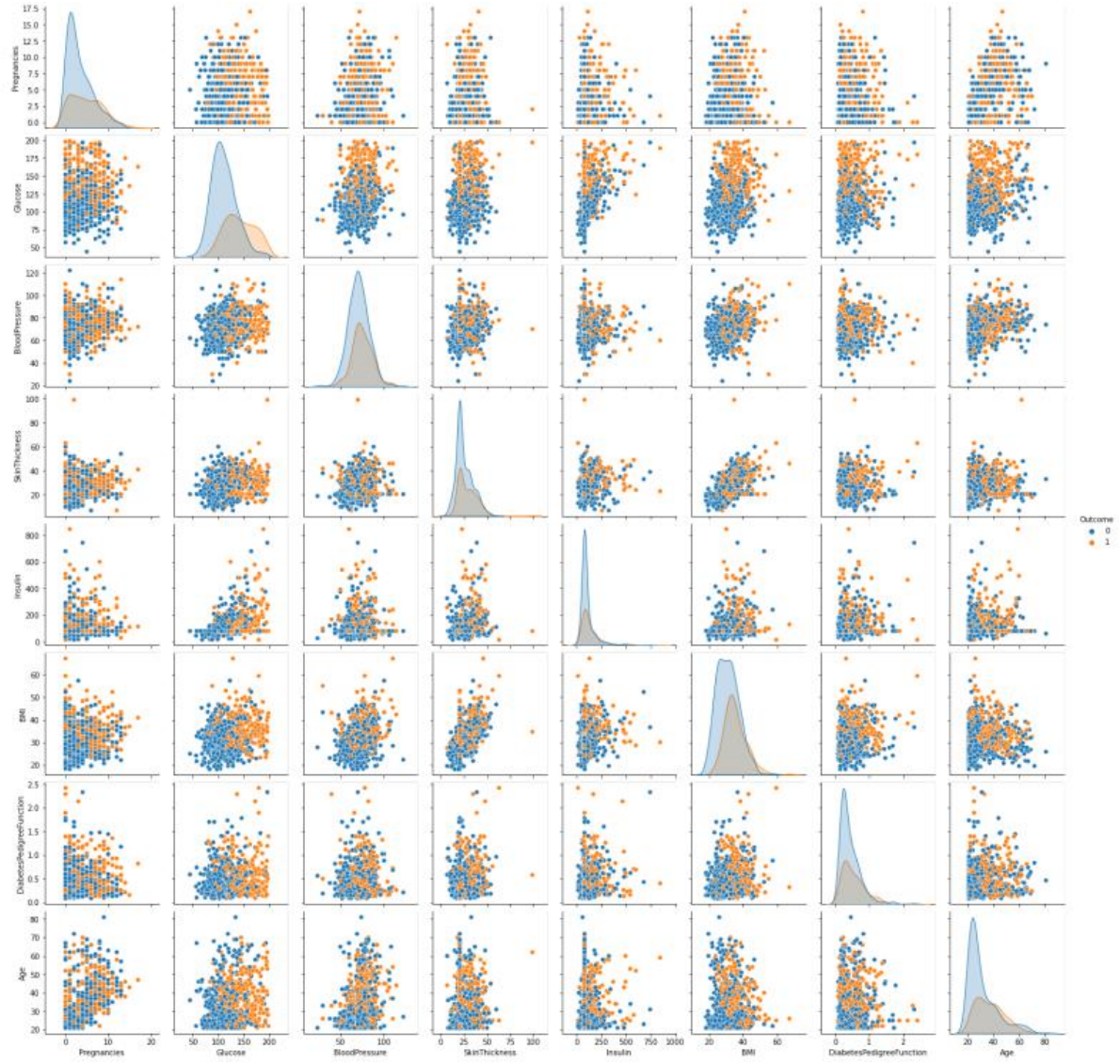
The descriptive statistics can be displayed as follows:

```
diabetes_clean.describe()
```

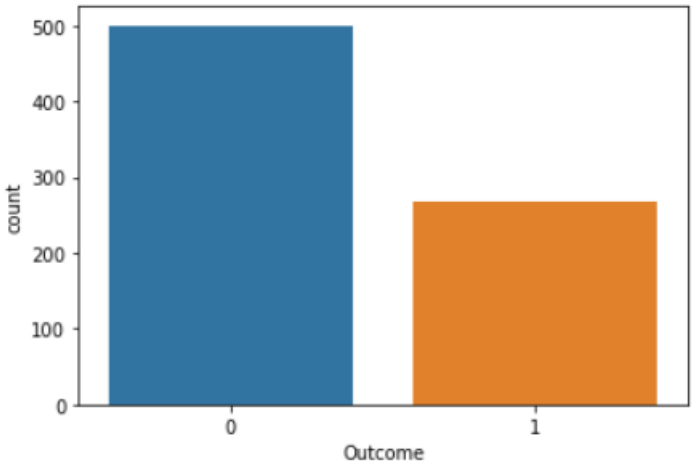
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	121.681605	72.254807	26.606479	118.660163	32.450805	0.471876
std	3.369578	30.436016	12.115932	9.631241	93.080358	6.875374	0.331329
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000
25%	1.000000	99.750000	64.000000	20.536458	79.799479	27.500000	0.243750
50%	3.000000	117.000000	72.000000	23.000000	79.799479	32.000000	0.372500
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000

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

```
<seaborn.axisgrid.PairGrid at 0x1278241cbb0>
```




```
sns.countplot(data = diabetes_clean, x = 'Outcome')
```



It highlights that there are more non-diabetic patients than diabetic patients.

```
plt.figure(figsize = [10, 10])
sns.heatmap(diabetes_clean.corr(), annot = True, fmt = '.3f', cmap = 'vlag_r', center = 0)
```

```
<AxesSubplot:>
```

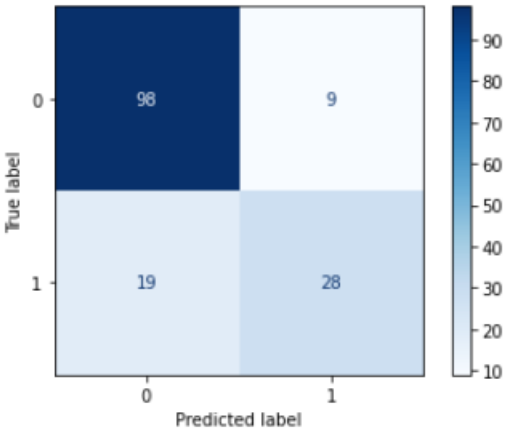
The above plot depicts that the attributes are not highly correlated with each other but themselves.

Model Fitting

```
# define X and y
y = diabetes_clean['Outcome']
X = diabetes_clean.drop('Outcome', axis=1)

# Splitting the data so 20% is for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

#feature scaling
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Logistic Regression

```
# Logistic Regression
# Fitting the model
LRmodel = LogisticRegression()
LRmodel.fit(X_train, y_train)
y_predict = LRmodel.predict(X_test)
#Confusion matrix
plot_confusion_matrix(LRmodel, X_test, y_test, cmap=plt.cm.Blues)
```



Here,

- 98 non-diabetic patients are correctly identified as non-diabetic (True Negative)
- 9 non-diabetic patients are incorrectly identified as diabetic (False Positive)
- 19 diabetic patients incorrectly identified as non-diabetic (False Negative)
- 28 diabetic patients are correctly identified as diabetic (True Positive)

```
print('F1 score: ',f1_score(y_predict,y_test)*100)
print('Accuracy: ',accuracy_score(y_predict,y_test)*100)
print('Precision score: ',precision_score(y_predict,y_test)*100)
print('Recall score: ',recall_score(y_predict,y_test)*100)
```

```
F1 score:  66.66666666666666
Accuracy:  81.81818181818183
Precision score:  59.57446808510638
Recall score:  75.67567567567568
```

Here,

- F1 Score is 67% indicating a well balance between precision and recall score.
- It is 82% accurate in predicting both diabetic and non-diabetic patients.
- It is 59% precised for not labelling a non-diabetic patient as diabetic.
- Recall Score is 76% which indicates it is sensitive for identifying diabetic patients among all the actual diabetic cases.

Decision Tree Classifier

```
#Decision tree classifier
DTmodel=DecisionTreeClassifier()
DTmodel.fit(X_train,y_train)
prediction=DTmodel.predict(X_test)
print('F1 score: ',f1_score(prediction,y_test)*100)
print('Accuracy: ',accuracy_score(prediction,y_test)*100)
print('Precision score: ',precision_score(prediction,y_test)*100)
print('Recall score: ',recall_score(prediction,y_test)*100)
```

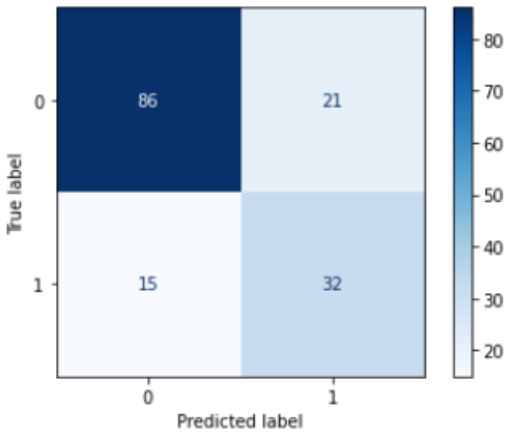
```
F1 score:  64.0
Accuracy:  76.62337662337663

Precision score:  68.08510638297872
Recall score:  60.37735849056604
```

Here,

- F1 Score is 64% indicating a well balance between precision and recall score.
- It is 77% accurate in predicting both diabetic and non-diabetic patients.
- It is 68% precised for not labelling a non-diabetic patient as diabetic.
- Recall Score is 60% which indicates it is sensitive for identifying diabetic patients among all the actual diabetic cases.

```
plot_confusion_matrix(DTmodel, X_test, y_test, cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x12785c70220>
```

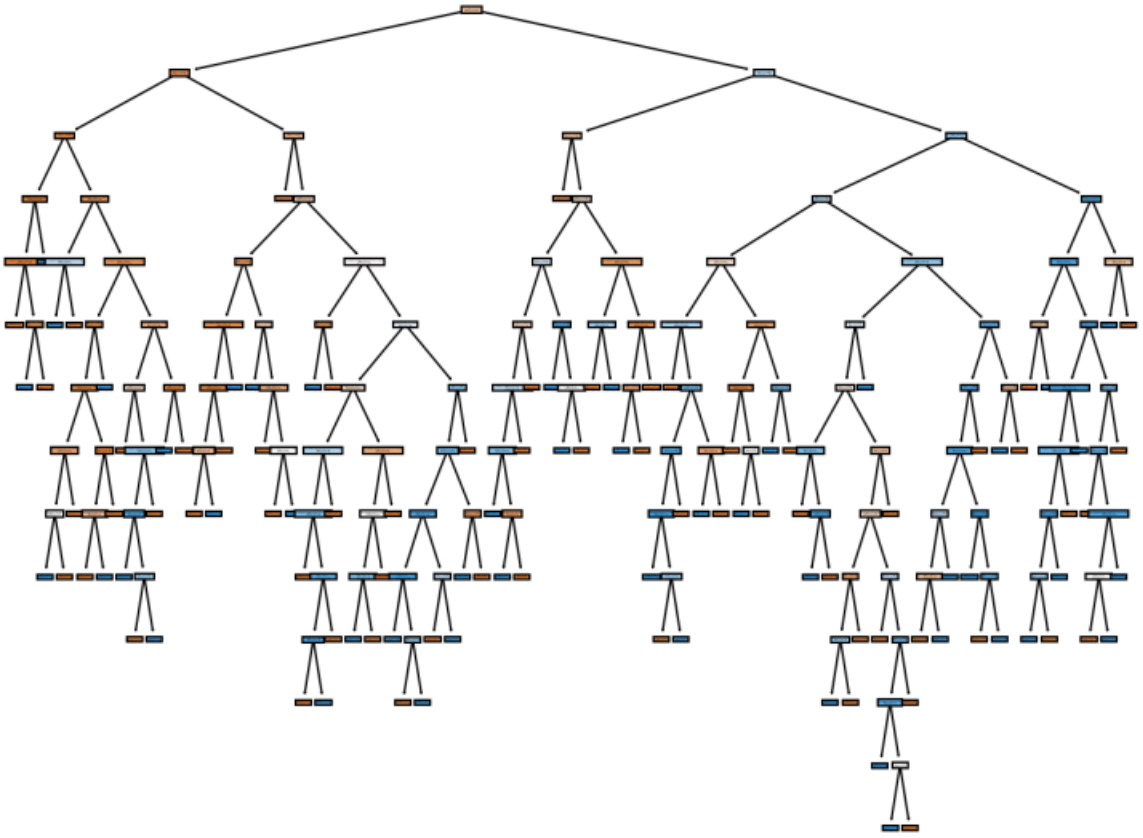


Here,

- 86 non-diabetic patients are correctly identified as non-diabetic (True Negative)
- 21 non-diabetic patients are incorrectly identified as diabetic (False Positive)
- 15 diabetic patients incorrectly identified as non-diabetic (False Negative)
- 32 diabetic patients are correctly identified as diabetic (True Positive)

Random Forest Classifier

```
from sklearn.tree import plot_tree
plt.figure(figsize=(10,8), dpi=150)
plot_tree(DTmodel, feature_names=X.columns, filled=True);
```



```
#Random forest classifier
RFmodel=RandomForestClassifier(n_estimators=100,random_state=0)
RFmodel.fit(X_train,y_train)
RF_prediction=RFmodel.predict(X_test)

print('F1 score: ',f1_score(RF_prediction,y_test)*100)
print('Accuracy: ',accuracy_score(RF_prediction,y_test)*100)
print('Precision score: ',precision_score(RF_prediction,y_test)*100)
print('Recall score: ',recall_score(RF_prediction,y_test)*100)
```

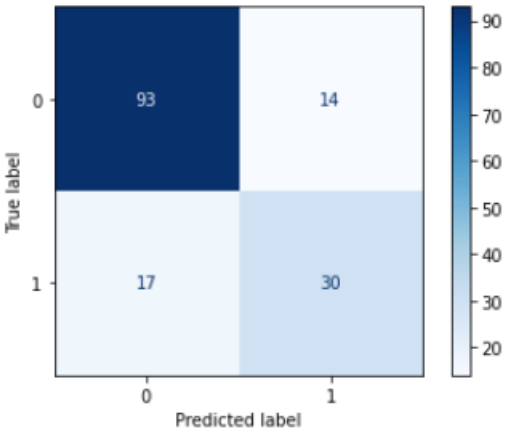
```
F1 score:  65.93406593406593
Accuracy:  79.87012987012987
Precision score:  63.829787234042556
Recall score:  68.18181818181817
```

Here,

- F1 Score is 66% indicating a well balance between precision and recall score.
- It is 79% accurate in predicting both diabetic and non-diabetic patients.
- It is 64% precised for not labelling a non-diabetic patient as diabetic.
- Recall Score is 68% which indicates it is sensitive for identifying diabetic patients among all the actual diabetic cases.

```
plot_confusion_matrix(RFmodel, X_test, y_test, cmap=plt.cm.Blues)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x127862a8bb0>
```



Here,

- 93 non-diabetic patients are correctly identified as non-diabetic (True Negative)
- 14 non-diabetic patients are incorrectly identified as diabetic (False Positive)
- 17 diabetic patients incorrectly identified as non-diabetic (False Negative)
- 30 diabetic patients are correctly identified as diabetic (True Positive)

Result

Since we need output sensitive predictions so it can be taken into account if a non-diabetic person is labeled as diabetic but a diabetic person should not be labeled as non-diabetic,so, as the cost of false positives and false negatives are very different ,we will prefer the model with highest F1 score and recall score. The logistic regression model has the highest F1 score and recall score, so we will consider that model for predictions.

Reference

- The dataset is taken from Kaggle.