DATA MINING: FINAL PROJECT

GROUP MEMBERS:

MALAIKA WAHEED (19I0726)

MESHAL CHEEMA (19I1977)

HAJIRA UZAIR (19I0737)

SUBMITTED TO:

DR. WASEEM SHAHZAD

SECTION: E

DIABETIES PREDICTION

USING MACHINE LEARNING ALGORITHMS

DIABETIES DATASET

The dataset chosen for the project is taken from Pima Indians diabetes dataset, which is originally from National Institute of Diabetes and Digestive and Kidney Diseases. The dataset consists of 8 independent and 1 dependent/target variable ("Outcome"). The rest variables are subjected to **cleaning**, **noise removal**, **feature selection** and at last **four machine learning algorithms** are applied which then predict whether the person has diabetes or not along with their respective accuracies.

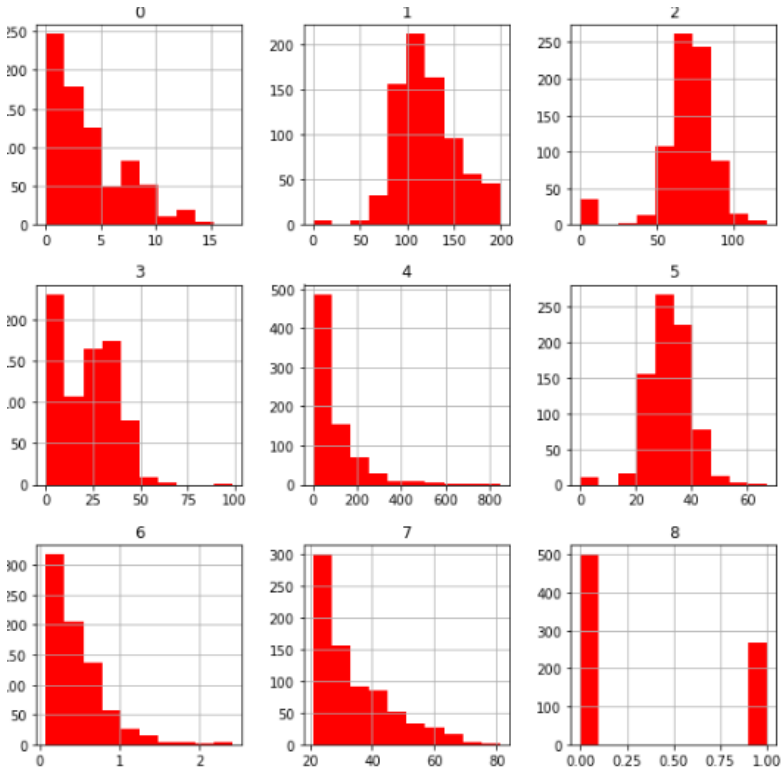
MACHINE LEARNING ALGORITHMS USED:

1) KNN (to predict diabetes)
2) Naïve Bayes (to predict diabetes)
3) Decision Tress (to predict diabetes)
4) Artificial Neural Networks (to predict diabetes)
5) Confusion Matrix (To evaluate performance)
6) K fold Cross Validate (Validation for accuracy on other datasets)

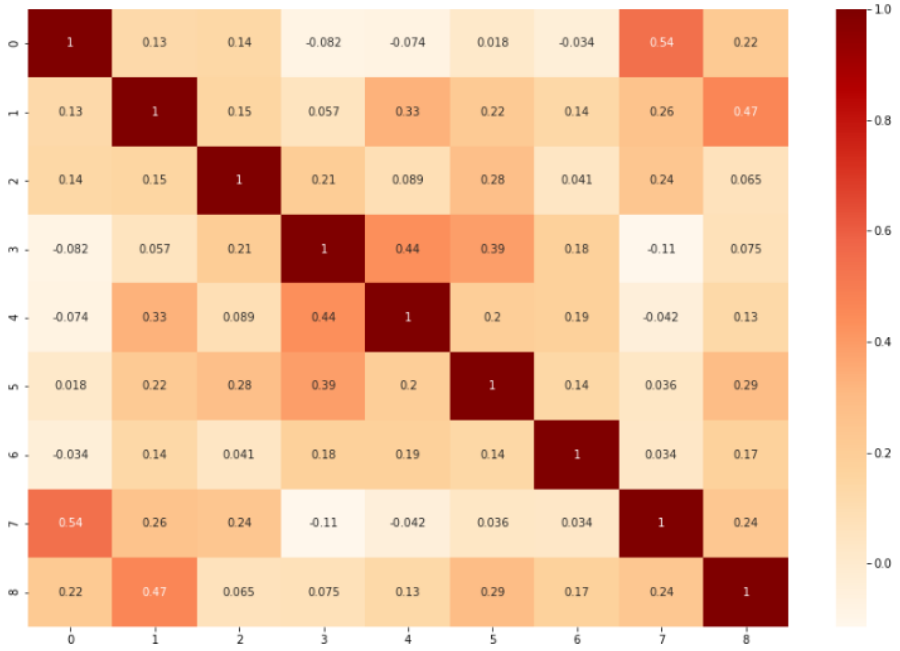
VIEW:

	0	1	2	3	4	5	6	7	8
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
...	...	...	...	...	...	...	...	...	...
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

HISTOGRAMS:



HEATMAP:



DATA SHAPE:

Rows: 768

Columns: 9 (Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes pedigree function, Outcome)

DATA CLEANING

The missing values function shows that there aren't any missing values in the dataset, so we need not to apply any method. Because all 9 Columns have all 768 values.

```
dataframe_Diabetes.isnull().sum()
```

```
0    0
1    0
2    0
3    0
4    0
5    0
6    0
7    0
8    0
dtype: int64
```

NOISE REMOVAL

Two techniques of noise removal are being shown. Normalization (Converting all values between 0 and 1) and discretization (dividing attributes into small continuous intervals mapped to some discrete variable). These preprocessing techniques will help in reducing redundancy and will handle continuous attributes

Discretization

	0	1	2	3	4	5	6	7
0	1.0	2.0	2.0	1.0	0.0	2.0	0.0	1.0
1	0.0	1.0	2.0	1.0	0.0	1.0	0.0	0.0
2	1.0	3.0	2.0	0.0	0.0	1.0	1.0	0.0
3	0.0	1.0	2.0	0.0	0.0	1.0	0.0	0.0
4	0.0	2.0	1.0	1.0	0.0	2.0	3.0	0.0
...	...	...	...	...	...	...	...	...
763	2.0	2.0	2.0	1.0	0.0	1.0	0.0	2.0
764	0.0	2.0	2.0	1.0	0.0	2.0	0.0	0.0
765	1.0	2.0	2.0	0.0	0.0	1.0	0.0	0.0
766	0.0	2.0	1.0	0.0	0.0	1.0	0.0	1.0
767	0.0	1.0	2.0	1.0	0.0	1.0	0.0	0.0

Normalization:

```
array([[0.35294118, 0.74371859, 0.59016393, ..., 0.23441503, 0.48333333,
        1.        ],
       [0.05882353, 0.42713568, 0.54098361, ..., 0.11656704, 0.16666667,
        0.        ],
       [0.47058824, 0.91959799, 0.52459016, ..., 0.25362938, 0.18333333,
        1.        ],
       ...,
       [0.29411765, 0.6080402 , 0.59016393, ..., 0.07130658, 0.15      ,
        0.        ],
       [0.05882353, 0.63316583, 0.49180328, ..., 0.11571307, 0.43333333,
        1.        ],
       [0.05882353, 0.46733668, 0.57377049, ..., 0.10119556, 0.03333333,
        0.        ]])
```

***** Normalized data has been used in the algorithms for predicting results and accuracies not discretized*****

FEATURE SELECTION

In real world, data is inconsistent and dirty. So, Feature Selection is important. More Features less accuracy because mostly features don not corelate with each other and results in more NN weights and decision tree nodes which then affects overall accuracy. So, by using LassoCV we select the features/attributes which are important to us and drop the ones which are unimportant.

Higher the alpha more features subjected to 0.

Results:

```
Best alpha Diabetes using built-in LassoCV: 0.004040
Best score using built-in LassoCV: 0.248383

Lasso picked 6 variables and discarded the other 2 variables
```

1) ALGORITHM: KNN

What is KNN? & How It works?

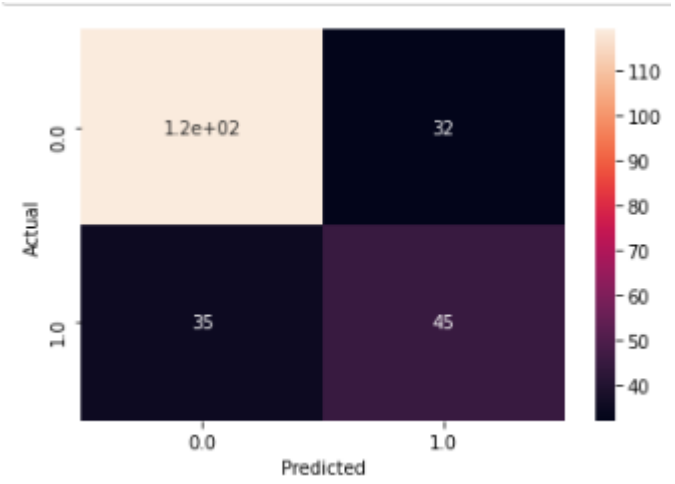
KNN, K-nearest neighbor (Classification Model) takes the difference of some target class with the dataset classes, and we find K most minimum distances i.e. predict our result based on the outcome of the classes which have minimum distance with our target class.

KNN & Diabetes

KNN is one of the most accurate classification and predictive algorithm. We are being given a health matrix in our dataset. Through that we predict each data point belongs to which group. It

is a binary classification problem. We have two values as our outcome 1 or 0 i.e., a person is diabetic or not respectively. So, we split our data into X and Y. X consists all of the independent variables and Y is the dependent variable. Distance Used in KNN is Euclidean but it can also be Manhattan or Minkowski, the values of K chosen is 7.

Confusion Matrix



Accuracy Of KNN

	precision	recall	f1-score	support
0.0	0.79	0.77	0.78	154
1.0	0.56	0.58	0.57	77
accuracy			0.71	231
macro avg	0.68	0.68	0.68	231
weighted avg	0.71	0.71	0.71	231

Accuracy 70.995670995671

Accuracy : 71%

2) ALGORITHM: NAÏVE BAYES

What is Naïve Bayes? & How It works?

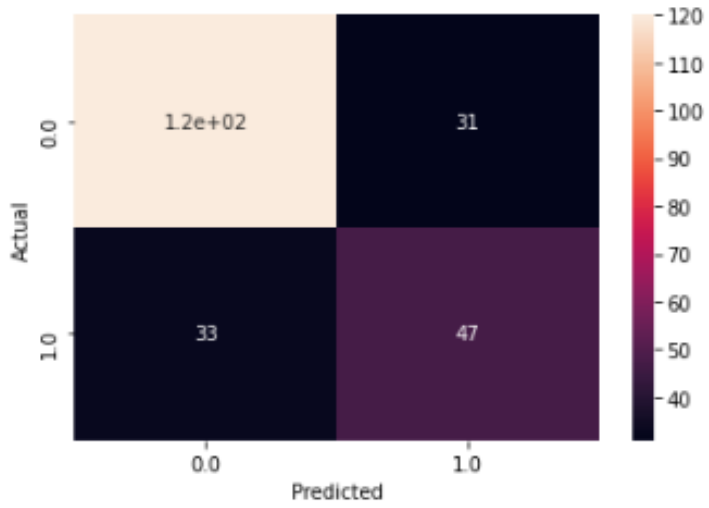
Naive Bayes generates classification models which assign labels (Supervised Learning), labels that are derived from some dataset. Such classification isn't associated with some algorithmic

rule. All Naive Bayes classifiers assume that the worth of 1 explicit attribute is independent of the properties of other attribute, given which is a category variable.

Naïve Bayes & Diabetes

Naive Bayes classifier approaches the features of diabetes dataset based on probability independently. It basically works on the Bayesian model of probability. For the estimation of any feature, it uses a method of maximum outcome which works independently and hence gives a probability of occurrence of a particular 1 or 0 situation as in our case. It works very efficiently in supervised learning.

Confusion Matrix



Accuracy Of Naïve Bayes

	precision	recall	f1-score	support
0.0	0.79	0.77	0.78	154
1.0	0.56	0.58	0.57	77
accuracy			0.71	231
macro avg	0.68	0.68	0.68	231
weighted avg	0.71	0.71	0.71	231

```
accuracy_score(y_test, y_pred)*100
```

```
72.2943722943723
```

Accuracy : 72%

3) DECISION TREES

What is Decision Tree? & How It works?

Decision trees are useful in interpretation and are represented graphically. To predict diabetes in a sample of n patients with p attributes, divide patients into j spaces. We will then have j non-overlapping distinct regions

$$R1, R2, R3\ldots\ldots Rj$$

The tree follows a top-down recursive binary splitting. It starts from a root node where all the observations are in one space which is then split into sub-branches until the stopping criterion is reached. The point of splitting in each successive sub-branch is decided using the greedy approach because it picks the best splitting point. The best splitting point is decided by calculating the entropy of each node. The split with the lowest entropy is selected. The method used for Entropy calculation is

$$-\sum_{i=1}^{c} P(x_i) log_b P(x_i)$$

Decision Tree & Diabetes

The dataset is split into training and test sets. Using the DecisionTreeClassifier class (Scikit-learn library) dataset is trained. Next, we make a forecast, get an "accuracy estimate", a classification report, and an error matrix.

Accuracy of Decision Tree

```
from sklearn.metrics import confus
print(accuracy_score(y_test, y_pre
```

```
75.97402597402598
```

Accuracy : 76 %

4) NEURAL NETWORKS

What is Neural Network? & How It works?

Neural networks are **a** set of algorithms, designed on similar pattern of how the human brain works, they are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input.

Neural Network & Diabetes

We divide our dataset in an output layer which gives us an output layer, between which there are many hidden layers. We check for the backward phase and forward phase and then update weights every time. So, for a given input features we develop one output feature that is outcome in our case.

Accuracy of Neural Network

```
    y_list = model.predict(x)
    y_list = np.around(y_list)
    y_list = np.asarray(y_list)
    if i == 0:
        print(accuracy_score(y, y_
```

```
74.487895716946
```

Accuracy : 74.4%

K FOLD CROSS VALIDATION

In this method the available data is randomly divided into k disjoint subsets of approximately equal size. One of the subsets is then used as the test set and the remaining k-1 sets are used for building the classifier. The test set is then used to estimate the accuracy. This is done repeatedly k times so that each subset is used as a test subset once. Then mean is calculated of all the k estimates.

FOR KNN

```
cv = KFold(n_splits=2)
scores = cross_val_score(neigh, df_di_X.values
print(scores*100)
```

```
[69.79166667 75.26041667]
```

FOR NAIVE BAYES

```
scores = cross_val_score(gnb, df_di_X.values,
print(scores*100)
```

```
[71.09375    74.73958333]
```

CONFUSION MATRIX

A “confusion matrix” is used to represent the results. The advantage of using this matrix is that it not only tells us how many got misclassified but also what misclassifications occurred. When there are two classes, positive (+) and negative (-), the confusion matrix consists of four cells, i.e., TP, FP, FN, and TN.