Machine Learning, Spring 2023: Project 1

NIKHIL DUGYALA

PIMA DIABETES DATASET

I have used the NumPy , matplotlib and Pandas packages, and I have used the Python programming language to implement this project.

**Dataset details**: The Pima dataset contains 768 observations and 8 features that describe medical details of each observation. The dataset includes diagnostic measurements for diabetes, such as pregnancy, glucose, blood pressure, skin thickness, insulin, BMI, and diabetes pedigree function, along with a binary label indicating whether the individual was diagnosed with diabetes or not. The goal is to predict whether a patient has diabetes based on these features.

Chart, box and whisker chart

Description automatically generatedChart, treemap chart

Description automatically generated

Histogram of the dataset Correlation Heatmap

Histogram will give us an idea of the range and spread of each feature. From these plots, we can see that there are some correlations between certain features, such as Age and Pregnancies, and BMI and SkinThickness.

Overall, these visualizations give us a better understanding of the data and its distribution, and can help inform our choice of algorithm and preprocessing steps.

I have done the training/test split by choosing the 80% of data for training and 20% of data for testing because it provides a good balance between having enough data to train a model effectively and having enough data to test the model's performance.

Diagram

Description automatically generated

Scatter Plot of the data set

**Algorithm Description:**

Data Pre-processing: In the data set columns such as ['Glucose','SkinThickness','BMI','Insulin','BloodPressure'] are having zero values which indicates missing data, so I have replaced the 0 values in the columns with the mean value of the non-zero values in the same column.

Feature Scaling: I have scaled the dataset by subtracting the mean and dividing by the standard deviation to ensure that all features have a similar scale and distribution, which can help the algorithm converge faster and more reliably.

Distance Metrics: I have used the three distance metrics such as Euclidean distance, Manhattan distance and Cosine similarity to measure the distances between datapoints.

**Algorithm Results:**

Below are the algorithm results which I have received by using three different distance metrics for K=11 in KNN algorithm:

Text

Description automatically generatedText

Description automatically generated

**Manhattan Distance Result** **Euclidean Distance Result**

Text

Description automatically generated Text

Description automatically generated

**Cosine Similarity output Minkowski distance with p=3**

Euclidean and Manhattan distances gave the same accuracy for the KNN algorithm with k=11 and accuracy for cosine similarity is low. In Minkowski distance p is the order of the Minkowski distance metric. When p=1, the distance metric is the Manhattan distance, and when p=2, the distance metric is the Euclidean distance.

Chart

Description automatically generated

**Runtime:**

The runtime of the KNN algorithm depends on the size of the dataset and the value of k chosen.

Text

Description automatically generated with low confidence

The actual "wall-clock" time that it took to compute the results is 51.89 seconds.

Text

Description automatically generated