COMP809 – Data Mining and Machine Learning

Lab 9: KNN Classification

In this lab, a detailed implementation of the K-Nearest Neighbours (KNN) classification algorithm using a real-world dataset is provided. Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot effectively use the insulin it produces. Insulin is a hormone that regulates blood sugar. Hyperglycaemia, or raised blood sugar, is a common effect of uncontrolled diabetes and over time leads to serious damage to many of the body's systems, especially the nerves and blood vessels.
The Diabetes dataset used in this lab holds data from female patients at least 21 years old of Pima Indian heritage residing near Phoenix, Arizona. More information can be found in UCI Machine Learning Repository.

Download Files

Download lab 9 files; Diabetes dataset (diabetesdata.csv) and the python code (COMP809_Lab9_KNN.ipynb). The dataset has been preprocessed (i.e. cleaned, and some missing values are imputed).

Install Libraries

for data science tasks (machine learning extensions) library. Mlxtend is a Python library of useful tools

```
pip install mlxtend
```

Import Libraries

```
#Import libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from mlxtend.plotting import plot_decision_regions
#visulaization modules

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
```

Create a DataFrame

```
diabetes = pd.read_csv('diabetesdata.csv')
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35	125	33.6	0.627	50	1
1	1	85.0	66.0	29	125	26.6	0.351	31	0
2	8	183.0	64.0	29	125	23.3	0.672	32	1
3	1	89.0	66.0	23	94	28.1	0.167	21	0
4	0	137.0	40.0	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48	180	32.9	0.171	63	0
764	2	122.0	70.0	27	125	36.8	0.340	27	0
765	5	121.0	72.0	23	112	26.2	0.245	30	0
766	1	126.0	60.0	29	125	30.1	0.349	47	1
767	1	93.0	70.0	31	125	30.4	0.315	23	0

768 rows × 9 columns

Figure 1: Features presented in the 'diabetesdata' dataset.

There are 768 instances in the dataset. The dataset holds the following 9 attributes:

1. Number of times pregnant
2. Plasma glucose concentration 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure in mm Hg
4. Triceps skin fold thickness in mm
5. 2-Hour serum insulin in mu U/ml
6. Body mass index measured as weight in kg/(height in m)^2
7. Diabetes pedigree function
8. Age in years
9. Outcome (1:diabetic or 0:healthy) (class)

Create the KNN Classifier

1. Explore the Data

Perform initial data exploration through Tasks 1 and 2.

Task 1: Create a Pearson heat-map to explore the correlation between the inputs (the first 8 columns) and the class (Outcome). Explain your findings.

Task 2: Check the class (outcome) distribution by plotting the count of outcomes by their value. Explain your findings.

2. Rescaling the Data

Looking at your data, you will notice that the data in each column has different ranges. It is recommended to bring all the features to the same scale when applying distance-based algorithms such as KNN. Rescaling brings all the features to the same scale.

Task 3: Rescale your dataset and define the Input and Target (class) variables.

3. Train and Test Splits with Stratification

Split your dataset to train (70%) and test sets. The test set holds data to be used for evaluating the model that has been created using the train set. Stratification helps to accommodate the problem of the 'imbalanced' data. The 'stratify parameter' in train_test_split() will ensure that both training and

test sets have the same proportions of the class labels.

Task 4: Split your dataset using sklearn's train_test_split(), and set the stratify parameter to accommodate the problem of 'imbalanced' data.

4. Create the Baseline KNN Classifier

Task 5: Fit a KNN classifier for a range of K (1:15) using the train set and save the scores. What is the max train score?

5. Presenting the Scores

Task 6: Plot the train and test score for each k. What is the best k? Explain your findings.

6. Create the Final Model Using the best K

Task 7: Setup a KNN classifier with the best k neighbours obtained in Task 6

7. Model Performance Analysis

Task 8: Generate the confusion matrix to summarize the final model performance.

Task 9: Provide the classification report (precision, recall, f1-score)

Self-directed Tasks

- Perform 10-fold cross validation to find the best number of neighbours.
- Study Grid search for hyper parameter tuning. In KNN classifier the parameter to be tuned is number of neighbours.