**kNN model to predict Diabetes**

**Introduction**

Diabetes is one of the most serious health challenges in both developing and developed countries. It has become leading cause of death. Detection of diabetes with optimal cost and better performance is the need of the age. The k-Nearest-Neighbours (kNN) is a simple but effective method for classification. I am using kNN classification model to predict the diabetes. Data cleaning is fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. I am using python functions to fix incorrect data. Medical data is multidimensional; hence data pre-processing step is applied to high dimensional data. In this experiment, I have used Pearson correlation between features and outcome to select best features to determine accuracy. Feature Transformation is a technique by which we can boost our model performance. I have used Scikit –learn preprocessing library functions for feature transformation. Then performance of each feature transformation function is validated using Accuracy for different kNN values.

**Methodology**

I am using KNN classification model to predict the diabetes. To get the best accuracy I am following four steps which are, data cleaning, feature selection, applying kNN model and feature transformation.

**Data Cleaning**: After analyzing the dataset, I found that following columns have an invalid zero values which is practically not possible.

1. Glucose
2. BloodPressure
3. SkinThickness
4. Insulin
5. BMI

It’s not possible that living human can have 0 glucose level or blood pressure etc. So, its look like these are missing values in dataset.

I replaced the 0 values with the mean/median value of respective column. e.g., in glucose column, I replaced the 0 values with the mean of glucose data.

**Feature Selection:**Feature selection, as a dimensionality reduction. technique, aims to choose a small subset of the. relevant features from the original ones by re- moving irrelevant, redundant, or noisy features

I am using Pearson correlation coefficient to determine correlation between input features and outcome. The value of Pearson's Correlation Coefficient can be between -1 to +1. 1 means that they are highly correlated and 0 means no correlation.

Heatmap function from seaborn library can be used to give 2-D representation of correlation between different features. Based on correlation value we can determine important features for the model

**kNN model application:**K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. K-nearest neighbors (KNN) algorithm uses ‘feature similarity’ to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

Chart, scatter chart

Description automatically generated

As we can see in above example with k=5, the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

The most challenging factor in kNN is to determine value of k. In my experiment, I have used 1 to 10 range of values of ‘k’ over K fold cross validation to determine best value of k for this dataset. Again, for each k value of kNN , I tried 1 to 10 range of folds of K-fold cross validation.

**Feature transformation:**Feature transformation is a mathematical transformation in which we apply a mathematical formula to a particular column (feature) and transform the values, which are useful for our further analysis. It is a technique by which we can boost our model performance. It is also known as Feature Engineering, which creates new features from existing features that may help improve the model performance.

I have used below feature transformation functions from sklearn-preprocessing library in my experiment.

1. MinMax Scaler
2. Standard Scaler
3. Robust Scaler
4. Quantile Transformer Scaler
5. Power Transformer Scaler

**Data**

Pima Indian dataset consist of 9 attributes (8 predictor and 1 class label). It’s a representation of 8 characteristics of 768 women. This dataset comes under supervised binary classification. Various attributes of dataset are described here.

|  |  |
| --- | --- |
| **Atribute** | **Description** |
| Pregnancies | Number of times pregnant |
| Glucose | Glucose concentration(mg/dL) |
| Blood Pressure | Diastolic Blood Pressure (mm Hg) |
| Skin Thickness | Skin thickness in (mm) |
| Insulin | Insulin level(mm U/ml) |
| BMI | Body Mass Index(weight/height square) |
| DiabetesPedigreeFunction | DiabetesPedigreeFunction |
| Age | Age in years |
| Outcome | 1:diabetic, 0:nondiabetic |

**Result**

Below are results after applying different steps/methods from methodology section

**Data Cleaning:**The accuracy of kNN.py is 0.7162 which is modified to 0.724 after data cleaning.

**Feature Selection:**I received below correlation matrix representation using heatmap function for diabetes data.

Chart, treemap chart

Description automatically generated

Above matrix shows that Glucose, BMI and age are highly correlated to the output. Whereas, blood pressure, insulin, DiabetesPedigreeFunction, pregnancies, skinthickness are less correlated to the output. However, Glucose is highly correlated to insulin whereas BMI is highly correlated to skin thickness.

So as a part of feature selection, I dropped blood pressure, DiabetesPedigreeFunction and pregnancies features and selected Glucose, SkinThickness, Insulin, BMI and Age.

**kNN model application:**I calculated the accuracy of for k value of kNN for each fold. Below is the performance graph for k values ranging from 1 to 10.

Chart, scatter chart

Description automatically generated

*kNN performance(accuracy) for different folds*

As per the above performance graph, maximum accuracy is for k=10 which is 0.76695.

**Feature Transformation:**

1. **MinMax Scaler**

Below is performance graph for k values ranging from 1 to 10 for each K fold(1 to 10)

Chart, scatter chart

Description automatically generated

*Min-Max scalar performance for k values*

Maximum accuracy for this scalar is 0.7834 for k is equal to 9

1. **Standard Scalar**

Below is performance graph for k values ranging from 1 to 10 for each K fold(1 to 10)

Chart, scatter chart

Description automatically generated

*Standard scalar performance for k values*

Maximum accuracy for this scalar is 0.765625 for k is equal to 8

1. **Robust Scalar**

Below is performance graph for k values ranging from 1 to 10 for each K fold(1 to 10)

Chart, scatter chart

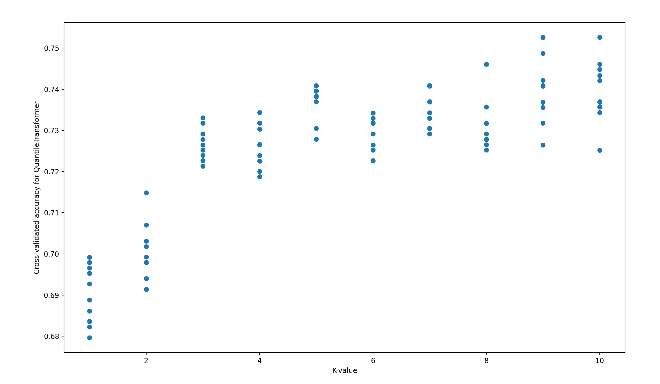
Description automatically generated

*Robust scalar performance for k values*

Maximum Accuracy for Robust Scalar is 0.7344 for K value 10

1. **Quantile Transformer**

Below is performance graph for k values ranging from 1 to 10 for each K fold(1 to 10)



*Quantile Transformer scalar performance for k values*

Maximum Accuracy for QuantileTransformer is 0.7526 for K value 10

1. **Power Transformer**

Below is performance graph for k values ranging from 1 to 10 for each K fold(1 to 10)

Chart, box and whisker chart

Description automatically generated

*Power Transformer scalar performance for k values*

Maximum Accuracy for Power Transformer is 0.7656 for K value 8.

**Discussion**

By analyzing above results we can say that accuracy of model is increased after data cleaning, kNN model and Feature engineering.

Initially without any changes the kNN.py file had accuracy of 0.7162 which is increased to 0.76695 after using feature selection and best K of kNN. The accuracy of model further increased by feature transformation.

We can compare the results of feature transformation in below table:

|  |  |  |
| --- | --- | --- |
| **Scalar Name** | **Maximum Accuracy** | **Best k value** |
| MinMaxScalar | 0.7839 | 9 |
| Standar Scalar | 0.7656 | 8 |
| Robust Scalar | 0.7344 | 10 |
| Quantile Trasformer | 0.7526 | 10 |
| Power Transformer | 0.7656 | 8 |

**Conclusion**

With reference to above results and discussion we can conclude that this model performs best with MinMaxScalar with k value as 9. The maximum accuracy of this model with MinMaxScalar is 0.7839

**References**

1. <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn>
2. <https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn>
3. <https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a>
4. <https://www.kaggle.com/uciml/pima-indians-diabetes-database>
5. <https://www.kaggle.com/amolbhivarkar/knn-for-classification-using-scikit-learn>
6. <https://www.kaggle.com/shrutimechlearn/step-by-step-diabetes-classification-knn-detailed>
7. <https://www.kaggle.com/pouryaayria/a-complete-ml-pipeline-tutorial-acu-86>