Introduction

A problem related to diagnostic prediction of the presence or absence of the Diabetics disease in patients were solved(experimented) using K Nearest Neighbor(kNN) classification [1] and different other techniques to do feature engineering, optimizing model generalization etc. The optimum strategy to solve the problem is determined.

Related work

Model building has been done using kNN for the the same dataset by many other previous developers using various method of model optimizing methods. I have referred their work to get more knowledge of the different techniques used in this experiment. [5], [6], [7].

Methodology

Throughout the experiment, Jupyter lab – Jupyter Notebook was used as the workspace. In addition, many predefined functions and tools were used from the scikit-learn libraries [16] and tools.

I. Data Understanding

Several pre-defined functions were used to understand the raw data in the given dataset. DataFrame.describe()[] was used to obtain descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values. Another function DataFrame.info() was used to obtain the count of NaN values in the dataset. There were no any NaN value in any of the columns.

II. Data Pre-processing

a. Three types of feature scaling available in scikit-learn library[16] were used such as MinMax, Robust and Standard to prevent the supervised learning models from getting biased toward a specific range of values. [8]
b. Feature Engineering using Sequential Feature Selection. Forward feature selection was used to determine the optimum combination of the features that results the best performance of the model. mlxtend.feature_selection libraby was used for this experiment [10]

III. Model fitting

This is a binary classification problem with data available for supervised learning. For this problem, k-Nearest Neighbor was used.

a. Determined the optimum number(k) of neighbors to run K Nearest Neighbor with Train-test Hold-out split using a basic python loop. To do the Train-test Hold-out split 40% of the whole dataset was used as the test data.
b. Determined the optimum number(k) of neighbors to run K Nearest Neighbor and the optimum number of k-folds to do the k-Fold Cross Validation using Hyperparameter tuning. A predefined function called GridSearch() was used in this experiment. [9]
c. Determined the most significant combination of predictors with the above found optimum no of k-folds and the no of k nearest neighbors.

d. The optimum strategy was decided based on the accuracy values of the built models in each experiement.

Data

The dataset used in this experiment is originally from the National Institute of Diabetes and Digestive and Kidney Diseases which is a part of the United States National Institutes of Health. The objective of the dataset is to diagnostically predict the absence or presence of the disease Diabetes in a patient, based on certain diagnostic measurements included in the dataset. It is mentioned in the source that several constraints were placed on the selection of these instances from a larger database such as all patients here are females at least 21 years old of Pima Indian heritage.[4]

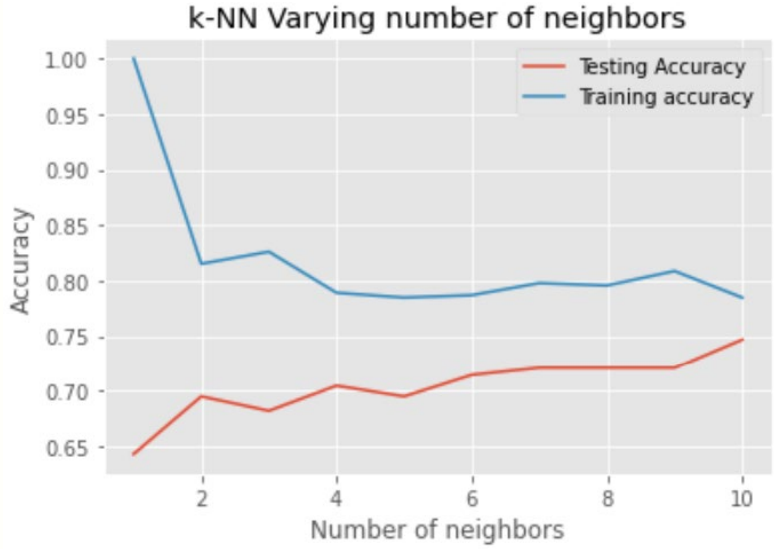
The dataset was used in csv file format and consisted of eight medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, glucose measure, blood pressure, skin thickness and diabetes pedigree function measure. The dataset used for this experiment includes 769 instances.

Result

I. Data Understanding – Summary statistics of the data set before processing.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

II. With Train-test Hold Out Validation the optimum number of neighbors to use was returned as 10.



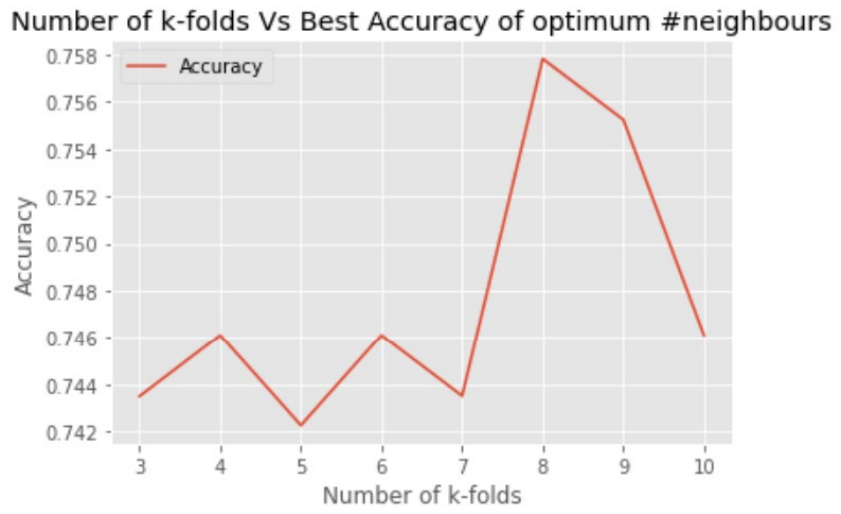

```
Max test score 74.67532467532467 % and k = [10]
```

III. With different feature scaling methods different Accuracy scores were obtained with kNN model fitting with 10 number of neighbors.

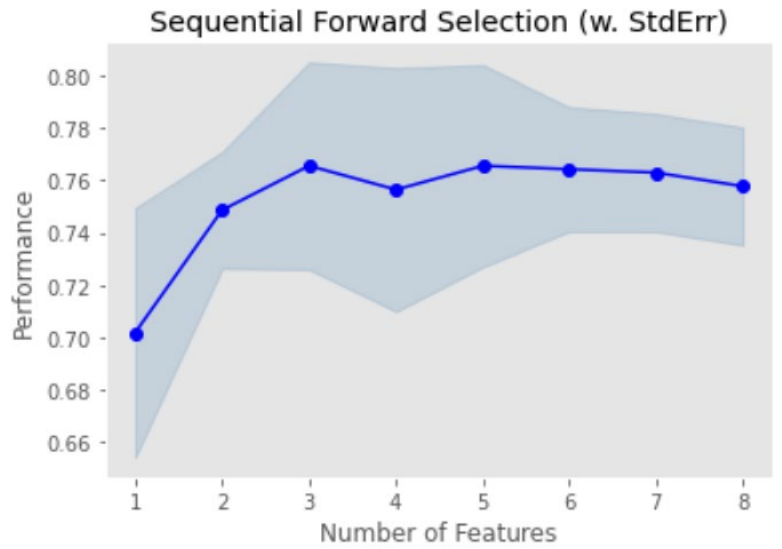
Feature Scaling method	Accuracy of the kNN model (neighbors = 10)
MinMax Scaler	0.7357609710550886
Robust Scaler	0.7409303115185468
Standard Scaler	0.7357609710550886

IV. From Hyperparameter tuning (using GridSearch() in scikit-learn), the optimum tradeoff between the number of kNN neighbors and the number of k-folds were obtained as 9 neighbors and 8 folds respectively. This combination results the highest accuracy which is 0.757812.

	k folds	best_score	n_neighbours
2	5	0.742254	3
0	3	0.743490	10
4	7	0.743501	8
1	4	0.746094	3
3	6	0.746094	3
7	10	0.746104	9
6	9	0.755236	9
5	8	0.757812	9



V. Feature selectin using Forward feature selection was applied using the results(best trade off of number of neighbors and the number of k-folds) of the previous experiment to find out the most effective combination of predictor features on the model accuracy. The most optimum number of features is 3 or 5 which had the highest accuracy score of 0.765625. The 3 features selected were ['Pregnancies', 'Glucose', 'BMI'] and the 5 features selected were ['Pregnancies', 'Glucose', 'BloodPressure', 'BMI', 'Age'].



VI. The best kNN model Accuracy obtained was 0.765625.

Cross Validation	Classifier	No of Features	Model Accuracy
Train-Test Hold Out Split	kNN(k=5)	8	0.6948051948051948
Train-Test Hold Out Split	kNN(k=10)	8	0.7207792207792207
K-fold Validation (kfolds=5)	kNN(k=10)	8	0.7513284101519396
K-fold Validation (kfolds=8)	kNN(k=9)	8	0.757812
K-fold Validation (kfolds=8)	kNN(k=9)	3,5	0.765625

Discussion

Using a larger test data size in Train-Test Hold Out validation results in higher accuracies. This could be due to the smaller number of instances in the dataset. Model accuracy is higher when k-Fold CV is performed in contract to Hold Out validation because cross validation using the whole dataset reduces the model overfitting and improves the generalization of the model. Feature engineering using Forward feature selection improves the model by eliminating irrelevant or less significant independent variables to the dependent variable. By removing irrelevant features or noise, the computational efficiency can be improved and the model's generalization error and be reduced. Accuracies when using Robust, MinMax and Standard scaler to transform the dataset changes with the number of neighbors. It needs to be further studied.

Conclusion

A kNN classifier with number of neighbors as 9 modeled with k-fold Cross Validation using 8 folds and 3,5 predictor features achieves the best score/accuracy of 0.765625 which is about 76%. This can be further improved.

Future work

The model can be improved using more sophisticated tools for feature engineering such as Feature extraction, Feature construction and Feature Transformation. The result of feature selection needs to be further investigated to find out what resulted in two combinations of predictor features and whether it can be further improved.

Reference

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
8. https://vitalflux.com/minmaxscaler-standardscaler-python-examples/#:~:text=The%20MinMaxscaler%20is%20a%20type,range%20from%20min%20to%20max.
9. https://stats.stackexchange.com/questions/207888/gridsearchcv-and-kfold
10. http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/
11. https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/

12. https://www.analyticsvidhya.com/blog/2020/10/a-comprehensive-guide-to-feature-selection-using-wrapper-methods-in-python/#:~:text=1.-,Forward%20selection,with%20all%20other%20remaining%20features.
13.
14. https://www.hindawi.com/journals/jam/2013/590614/
15. https://www.edwinwenink.xyz/posts/61-feature_preprocessing_and_train_test_leakage/
16. https://scikit-learn.org/stable/index.html