EXPERIMENT 9

Aim

To classify a dataset using KNN based classification algorithm

Software Used

Google Colab

Program Code and Output

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for data visualization purposes
import seaborn as sns # for data visualization
%matplotlib inline
```

```
[5] data = '/content/breast-cancer-wisconsin.data.csv'

    df = pd.read_csv(data, header=None)
```

```
[6] # view dimensions of dataset

    df.shape
```

```
(699, 11)
```

```
# preview the dataset

df.head()
```

	0	1	2	3	4	5	6	7	8	9	10
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

```
[8] col_names = ['Id', 'Clump_thickness', 'Uniformity_Cell_Size', 'Uniformity_Cell_Shape', 'Marginal_Adhesion',
                 'Single_Epithelial_Cell_Size', 'Bare_Nuclei', 'Bland_Chromatin', 'Normal_Nucleoli', 'Mitoses', 'Class']

    df.columns = col_names

    df.columns
```

```
Index(['Id', 'Clump_thickness', 'Uniformity_Cell_Size',
       'Uniformity_Cell_Shape', 'Marginal_Adhesion',
       'Single_Epithelial_Cell_Size', 'Bare_Nuclei', 'Bland_Chromatin',
       'Normal_Nucleoli', 'Mitoses', 'Class'],
      dtype='object')
```

```
[9] # let's agian preview the dataset

    df.head()
```

	Id	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin
0	1000025	5	1	1	1	2	1	3
1	1002945	5	4	4	5	7	10	3
2	1015425	3	1	1	1	2	2	3
3	1016277	6	8	8	1	3	4	3
4	1017023	4	1	1	3	2	1	3

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 10 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Clump_thickness              699 non-null    int64
 1   Uniformity_Cell_Size         699 non-null    int64
 2   Uniformity_Cell_Shape        699 non-null    int64
 3   Marginal_Adhesion            699 non-null    int64
 4   Single_Epithelial_Cell_Size  699 non-null    int64
 5   Bare_Nuclei                  699 non-null    object
 6   Bland_Chromatin              699 non-null    int64
 7   Normal_Nucleoli              699 non-null    int64
 8   Mitoses                      699 non-null    int64
 9   Class                        699 non-null    int64
dtypes: int64(9), object(1)
memory usage: 54.7+ KB
```

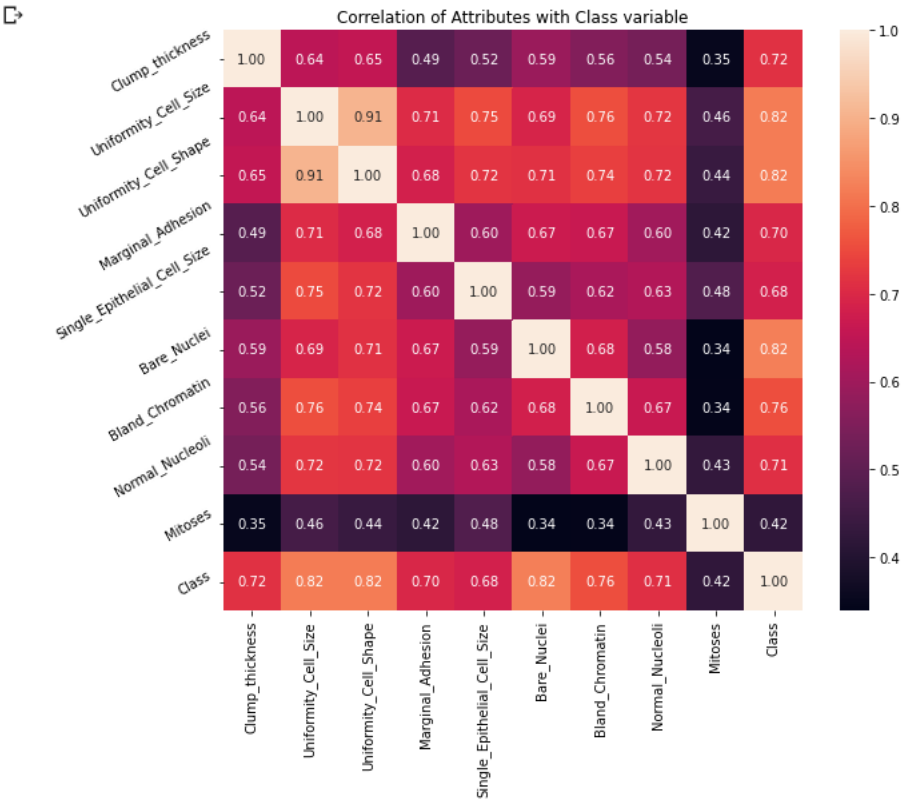
```
[13] df['Bare_Nuclei'] = pd.to_numeric(df['Bare_Nuclei'], errors='coerce')
```

```
# check missing values in variables

df.isnull().sum()
```

```
Clump_thickness                0
Uniformity_Cell_Size           0
Uniformity_Cell_Shape          0
Marginal_Adhesion              0
Single_Epithelial_Cell_Size    0
Bare_Nuclei                   16
Bland_Chromatin                0
Normal_Nucleoli                0
Mitoses                        0
Class                          0
dtype: int64
```

```
[26] plt.figure(figsize=(10,8))
     plt.title('Correlation of Attributes with Class variable')
     a = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white')
     a.set_xticklabels(a.get_xticklabels(), rotation=90)
     a.set_yticklabels(a.get_yticklabels(), rotation=30)
     plt.show()
```




```
[27] X = df.drop(['Class'], axis=1)

     y = df['Class']
```

```
[28] # split X and y into training and testing sets

     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
[ ]  # check the shape of X_train and X_test

     X_train.shape, X_test.shape
```

```
     ((559, 9), (140, 9))
```

```
# impute missing values in X_train and X_test with respective column median in X_train
#print([x for x in [X_train,X_test]])
for df1 in [X_train, X_test]:
    for col in X_train.columns:
        col_median=X_train[col].median()
        df1[col].fillna(col_median, inplace=True)
```

```
# check again missing values in numerical variables in X_train

X_train.isnull().sum()
```

```
Clump_thickness               0
Uniformity_Cell_Size          0
Uniformity_Cell_Shape         0
Marginal_Adhesion             0
Single_Epithelial_Cell_Size   0
Bare_Nuclei                   0
Bland_Chromatin               0
Normal_Nucleoli               0
Mitoses                       0
dtype: int64
```

```
[39] cols = X_train.columns
```

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

```
[41] X_train = pd.DataFrame(X_train, columns=[cols])
```

```
[42] X_test = pd.DataFrame(X_test, columns=[cols])
```

```
# import KNeighbors ClaSSifier from sklearn
from sklearn.neighbors import KNeighborsClassifier


# instantiate the model
knn = KNeighborsClassifier(n_neighbors=3)


# fit the model to the training set
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

```
[45] y_pred = knn.predict(X_test)

     y_pred
```

```
array([2, 2, 4, 2, 4, 2, 4, 2, 4, 2, 2, 2, 4, 4, 4, 2, 2, 4, 4, 2, 4, 4,
       2, 2, 2, 4, 2, 2, 4, 4, 2, 2, 2, 2, 2, 2, 2, 4, 2, 2, 2, 2, 2, 2,
       4, 4, 2, 4, 2, 4, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 4, 2, 2, 4, 4, 4,
       4, 2, 2, 4, 2, 2, 4, 4, 2, 2, 2, 2, 4, 2, 2, 2, 4, 2, 2, 2, 4, 2,
       4, 4, 2, 2, 2, 4, 2, 2, 2, 4, 2, 4, 4, 2, 2, 2, 4, 2, 2, 2, 2, 2,
       4, 4, 4, 2, 2, 2, 2, 2, 4, 4, 4, 4, 2, 4, 2, 2, 4, 4, 4, 4, 4, 2,
       2, 4, 4, 2, 2, 4, 2, 2])
```

```
[48] from sklearn.metrics import accuracy_score

     print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.9714
```

```
[49] y_pred_train = knn.predict(X_train)
```

```
[50] print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

```
Training-set accuracy score: 0.9821
```

```
[51] # print the scores on training and test set

     print('Training set score: {:.4f}'.format(knn.score(X_train, y_train)))

     print('Test set score: {:.4f}'.format(knn.score(X_test, y_test)))
```

```
Training set score: 0.9821
Test set score: 0.9714
```

```
# instantiate the model with k=5
knn_5 = KNeighborsClassifier(n_neighbors=5)


# fit the model to the training set
knn_5.fit(X_train, y_train)


# predict on the test-set
y_pred_5 = knn_5.predict(X_test)


print('Model accuracy score with k=5 : {0:0.4f}'. format(accuracy_score(y_test, y_pred_5)))
```

```
Model accuracy score with k=5 : 0.9714
```

```
# instantiate the model with k=6
knn_6 = KNeighborsClassifier(n_neighbors=6)


# fit the model to the training set
knn_6.fit(X_train, y_train)


# predict on the test-set
y_pred_6 = knn_6.predict(X_test)


print('Model accuracy score with k=6 : {0:0.4f}'. format(accuracy_score(y_test, y_pred_6)))
```

```
Model accuracy score with k=6 : 0.9786
```

Rebuild kNN Classification model using k=7

```
[54] # instantiate the model with k=7
knn_7 = KNeighborsClassifier(n_neighbors=7)


# fit the model to the training set
knn_7.fit(X_train, y_train)


# predict on the test-set
y_pred_7 = knn_7.predict(X_test)


print('Model accuracy score with k=7 : {0:0.4f}'. format(accuracy_score(y_test, y_pred_7)))
```

```
Model accuracy score with k=7 : 0.9786
```

Rebuild kNN Classification model using k=8

```
# instantiate the model with k=8
knn_8 = KNeighborsClassifier(n_neighbors=8)


# fit the model to the training set
knn_8.fit(X_train, y_train)


# predict on the test-set
y_pred_8 = knn_8.predict(X_test)


print('Model accuracy score with k=8 : {0:0.4f}'. format(accuracy_score(y_test, y_pred_8)))
```

```
Model accuracy score with k=8 : 0.9786
```

- Rebuild kNN Classification model using k=9

```
# instantiate the model with k=9
knn_9 = KNeighborsClassifier(n_neighbors=9)


# fit the model to the training set
knn_9.fit(X_train, y_train)


# predict on the test-set
y_pred_9 = knn_9.predict(X_test)


print('Model accuracy score with k=9 : {0:0.4f}'. format(accuracy_score(y_test, y_pred_9)))
```

```
Model accuracy score with k=9 : 0.9714
```

Discussion and Conclusion

K Nearest Neighbours classifier is implemented on a breast cancer dataset. Dataset is scaled using z score normalisation. Assumption that the data are missing completely at random (MCAR) is taken in case of Null values observed in the column 'Bare_Nuclei' of the dataset. Median imputation is used because it is robust to outliers as compared to mean imputation. Original model accuracy score with k=3 is 0.9714. Now, we can see that we get same accuracy score of 0.9714 with k=5. But, if we increase the value of k further, this will result in enhanced accuracy. With k=6,7,8 we get accuracy score of 0.9786. So, it results in performance improvement. If we increase k to 9, then accuracy decreases again to 0.9714.

CRITERIA	TOTAL MARKS	MARKS OBTAINED	COMMENTS
Concept (A)	2		
Implementation (B)	2		
Performance (C)	2		
Total	6		