

EXPERIMENT- 9

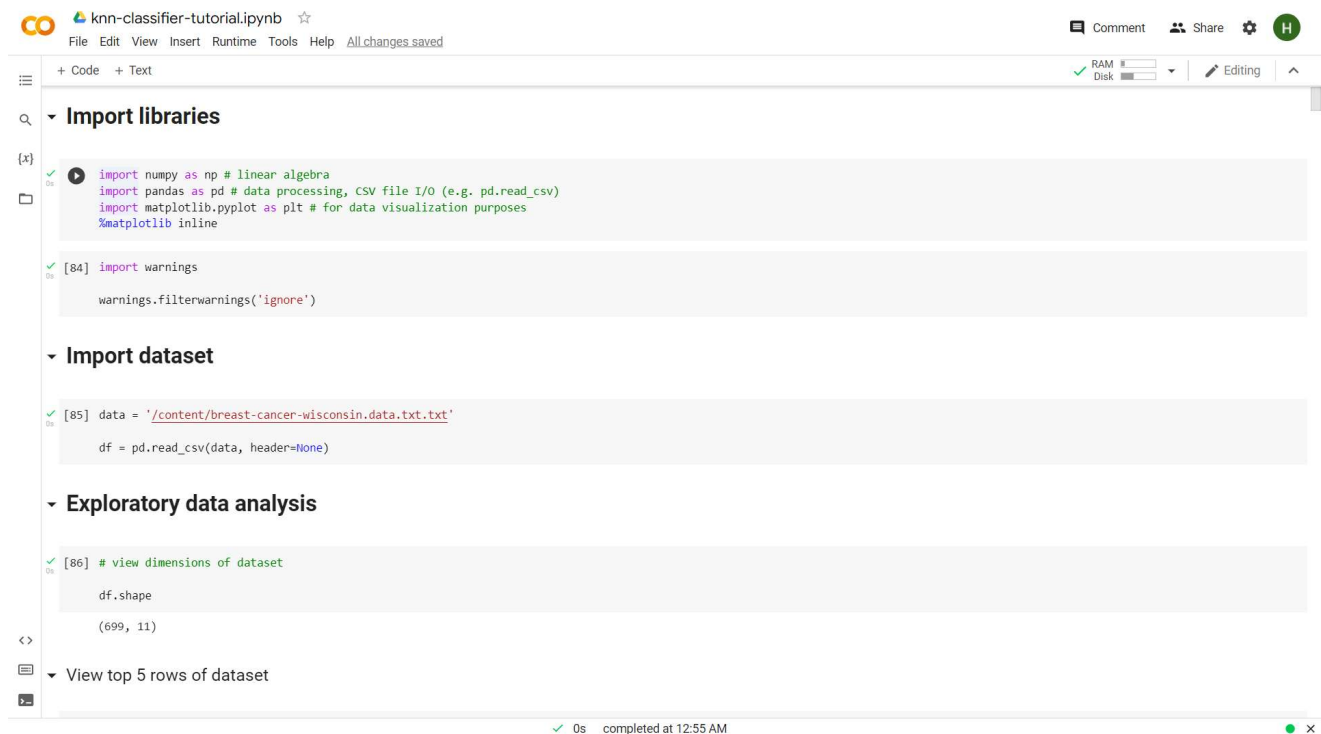
AIM

Implement a KNN based classification on a dataset.

SOFTWARE USED

Google Colab Platform - Python Programming Language

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
%matplotlib inline

[84] import warnings
     warnings.filterwarnings('ignore')

# Import dataset

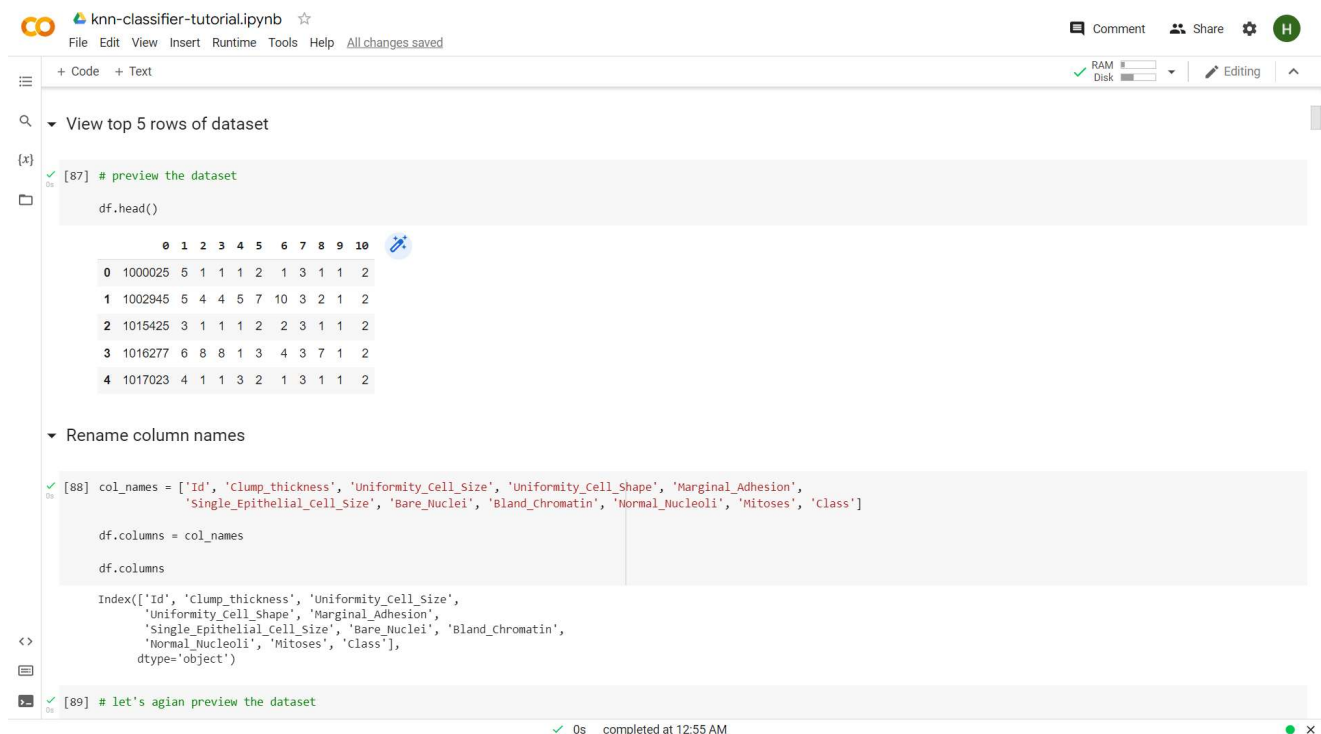
[85] data = '/content/breast-cancer-wisconsin.data.txt.txt'
     df = pd.read_csv(data, header=None)

# Exploratory data analysis

[86] # view dimensions of dataset
     df.shape

(699, 11)

# View top 5 rows of dataset
```



```
# View top 5 rows of dataset

[87] # 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

# Rename column names

[88] 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')

[89] # let's again preview the dataset
```

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

```

Normal_Nucleoli', 'Mitoses', 'Class'],
dtype='object')

```

[89] # let's again preview the dataset

```

df.head()

```

	Id	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoli	Mitoses	Class
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

Drop redundant columns

[90] # drop Id column from dataset

```

df.drop('Id', axis=1, inplace=True)

```

View summary of dataset

[91] # view summary of dataset

```

df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698

```

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

```

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

```

Frequency distribution of values in variables

[92] for var in df.columns:

```

    print(df[var].value_counts())

```

```

2    386
3     72
4     48
1     47
6     41
5     39
10    31
8     21
7     12
9      2
Name: Single_Epithelial_cell_Size, dtype: int64

```

0s completed at 12:55 AM

knnclassifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```

10 31
8 21
7 12
9 2
Name: Single_Epithelial_Cell_Size, dtype: int64
1 492
10 132
2 30
5 30
3 28
8 21
4 19
? 16
9 9
7 8
6 4
Name: Bare_Nuclei, dtype: int64
2 166
3 165
1 152
7 73
4 40
5 34
8 28
10 20
9 11
6 10
Name: Bland_Chromatin, dtype: int64
1 443
10 61
3 44
2 36
8 24
6 22
5 19
4 18
7 16
9 16
Name: Normal_Nucleoli, dtype: int64
1 579
2 35
3 33
10 14

```

0s completed at 12:55 AM

knnclassifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```

9 16
Name: Normal_Nucleoli, dtype: int64
1 579
2 35
3 33
10 14
4 12
7 9
8 8
5 6
6 3
Name: Mitoses, dtype: int64
2 458
4 241
Name: Class, dtype: int64

```

▼ Convert data type of Bare_Nuclei to integer

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

▼ Check data types of columns of dataframe

```
[94] df.dtypes
```

```

Clump_thickness      int64
Uniformity_Cell_Size int64
Uniformity_Cell_Shape int64
Marginal_Adhesion    int64
Single_Epithelial_Cell_Size int64
Bare_Nuclei          float64
Bland_Chromatin      int64
Normal_Nucleoli      int64
Mitoses              int64
Class                int64
dtype: object

```

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

RAM Disk Editing

+ Code + Text

Mitosesint64

Classint64

dtype: object

Explore problems within variables

Now, I will explore problems within variables.

Missing values in variables

[95] # check missing values in variables

df.isnull().sum()

Clump_thickness0

Uniformity_Cell_Size0

Uniformity_Cell_Shape0

Marginal_Adhesion0

Single_Epithelial_Cell_Size0

Bare_Nuclei16

Bland_Chromatin0

Normal_Nucleoli0

Mitoses0

Class0

dtype: int64

[96] # check 'na' values in the dataframe

df.isna().sum()

Clump_thickness0

Uniformity_Cell_Size0

Uniformity_Cell_Shape0

Marginal_Adhesion0

Single_Epithelial_Cell_Size0

Bare_Nuclei16

completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

RAM Disk Editing

+ Code + Text

Uniformity_Cell_Size0

Uniformity_Cell_Shape0

Marginal_Adhesion0

Single_Epithelial_Cell_Size0

Bare_Nuclei16

Bland_Chromatin0

Normal_Nucleoli0

Mitoses0

Class0

dtype: int64

[97] # check frequency distribution of 'Bare_Nuclei' column

df['Bare_Nuclei'].value_counts()

1.0402

10.0132

2.030

5.030

3.028

8.021

4.019

9.09

7.08

6.04

Name: Bare_Nuclei, dtype: int64

[98] # check unique values in 'Bare_Nuclei' column

df['Bare_Nuclei'].unique()

array([1., 10., 2., 4., 3., 9., 7., nan, 5., 8., 6.])

[99] # check for nan values in 'Bare_Nuclei' column

df['Bare_Nuclei'].isna().sum()

16

completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

df['Bare_Nuclei'].isna().sum()

16

check frequency distribution of target variable Class

[100] # view frequency distribution of values in 'Class' variable

df['Class'].value_counts()

2 458
4 241
Name: Class, dtype: int64

check percentage of frequency distribution of Class

[101] # view percentage of frequency distribution of values in 'Class' variable

df['Class'].value_counts()/np.float(len(df))

2 0.655222
4 0.344778
Name: Class, dtype: float64

Outliers in numerical variables

[102] # view summary statistics in numerical variables

print(round(df.describe(),2))

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	\
count	699.00	699.00	699.00	

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk

Editing

print(round(df.describe(),2))

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	\
count	699.00	699.00	699.00	
mean	4.42	3.13	3.21	
std	2.82	3.05	2.97	
min	1.00	1.00	1.00	
25%	2.00	1.00	1.00	
50%	4.00	1.00	1.00	
75%	6.00	5.00	5.00	
max	10.00	10.00	10.00	

	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	\
count	699.00	699.00	683.00	
mean	2.81	3.22	3.54	
std	2.86	2.21	3.64	
min	1.00	1.00	1.00	
25%	1.00	2.00	1.00	
50%	1.00	2.00	1.00	
75%	4.00	4.00	6.00	
max	10.00	10.00	10.00	

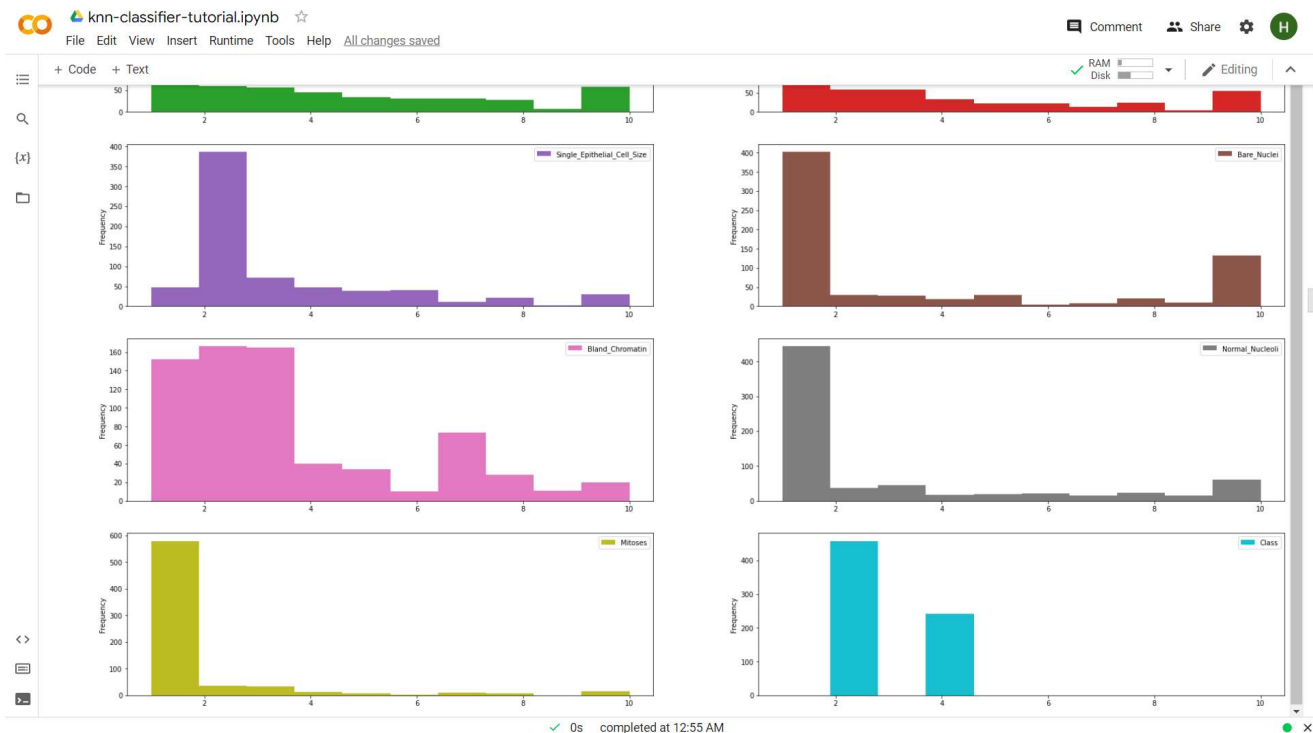
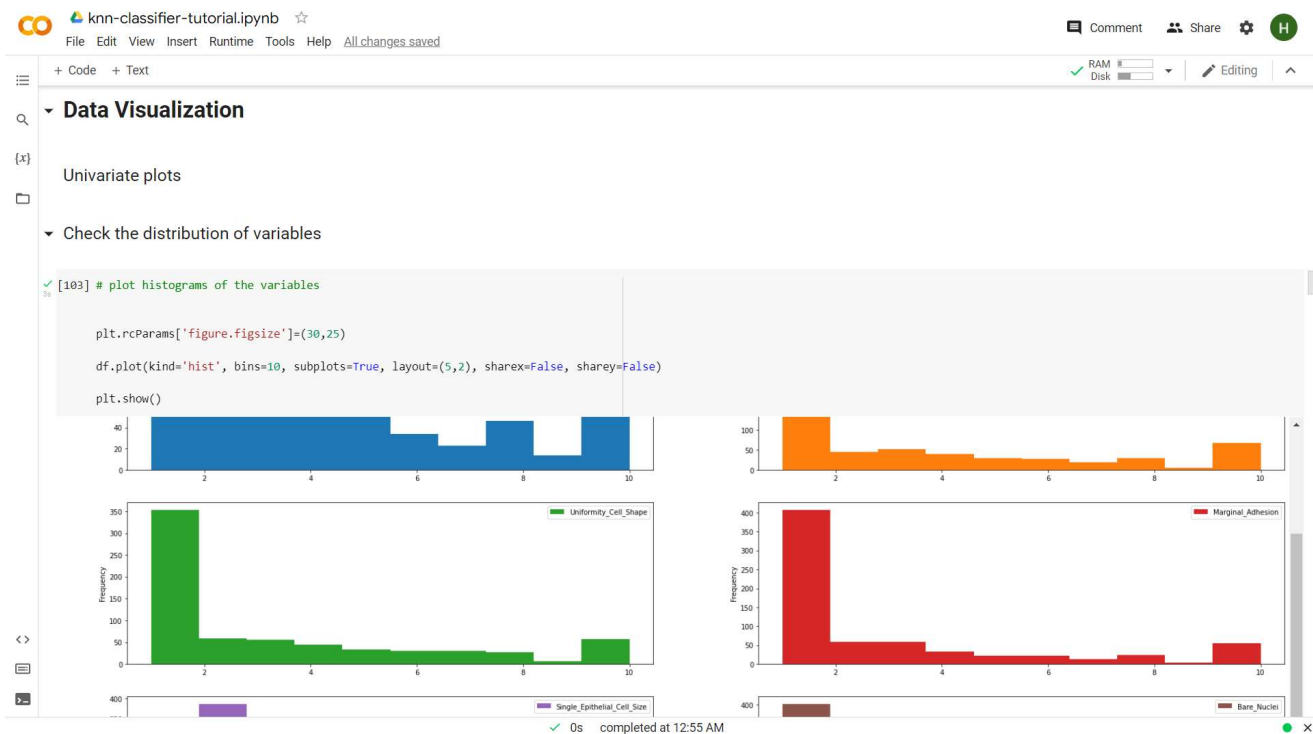
	Bland_Chromatin	Normal_Nucleoli	Mitoses	Class
count	699.00	699.00	699.00	699.00
mean	3.44	2.87	1.59	2.69
std	2.44	3.05	1.72	0.95
min	1.00	1.00	1.00	2.00
25%	2.00	1.00	1.00	2.00
50%	3.00	1.00	1.00	2.00
75%	5.00	4.00	1.00	4.00
max	10.00	10.00	10.00	4.00

KNN algorithm is robust to outliers.

Data Visualization

0s completed at 12:55 AM

67



knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

We can see that all the variables in the dataset are positively skewed.

Multivariate plots

Estimating correlation coefficients

```
[104] correlation = df.corr()
[105] correlation['Class'].sort_values(ascending=False)
```

	Class
Class	1.000000
Bare_Nuclei	0.822696
Uniformity_Cell_Shape	0.818934
Uniformity_Cell_Size	0.817904
Bland_Chromatin	0.756616
Clump_thickness	0.716001
Normal_Nucleoli	0.712244
Marginal_Adhesion	0.696800
Single_Epithelial_Cell_Size	0.682785
Mitoses	0.423170

Name: Class, dtype: float64

Discover patterns and relationships

Correlation Heat Map

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Editing

Correlation Heat Map

```
[106] 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()
```

Correlation of Attributes with Class variable

	Class
Clump_thickness	0.72
Uniformity_Cell_Size	0.82
Uniformity_Cell_Shape	0.82
Marginal_Adhesion	0.42
Single_Epithelial_Cell_Size	0.68
Bare_Nuclei	0.82
Bland_Chromatin	0.76
Normal_Nucleoli	0.71
Mitoses	0.42
Class	1.00

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

Class

Clump_thickness 0.72 0.82 0.82 0.70 0.68 0.82 0.76 0.71 0.42 1.00

Uniformity_Cell_Size

Uniformity_Cell_Shape

Marginal_Adhesion

Single_Epithelial_Cell_Size

Bare_Nuclei

Bland_Chromatin

Normal_Nucleoli

Mitoses

Class

0.4

▼ Declare feature vector and target variable

```
[107] X = df.drop(['class'], axis=1)
      y = df['class']
```

▼ Split data into separate training and test set

```
[108] # 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)
```

```
[109] # check the shape of X_train and X_test
      X_train.shape, X_test.shape
      ((559, 9), (140, 9))
```

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

X_train.shape, X_test.shape

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

▼ Feature Engineering

```
[110] # check data types in X_train
      X_train.dtypes
```

```
Clump_thickness      int64
Uniformity_Cell_Size int64
Uniformity_Cell_Shape int64
Marginal_Adhesion    int64
Single_Epithelial_Cell_Size int64
Bare_Nuclei          float64
Bland_Chromatin      int64
Normal_Nucleoli      int64
Mitoses              int64
dtype: object
```

▼ Engineering missing values in variables

```
[111] # check 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          13
Bland_Chromatin      0
```

0s completed at 12:55 AM

00 knn-classifier-tutorial.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```

Uniformity_Cell_Size      0
Uniformity_Cell_Shape     0
Marginal_Adhesion         0
Single_Epithelial_Cell_Size 0
Bare_Nuclei               13
Bland_Chromatin           0
Normal_Nucleoli           0
Mitoses                   0
dtype: int64

```

✓ [112] # check missing values in numerical variables in X_test

```

X_test.isnull().sum()

Clump_thickness      0
Uniformity_Cell_Size 0
Uniformity_Cell_Shape 0
Marginal_Adhesion    0
Single_Epithelial_Cell_Size 0
Bare_Nuclei          3
Bland_Chromatin      0
Normal_Nucleoli      0
Mitoses              0
dtype: int64

```

✓ [113] # print percentage of missing values in the numerical variables in training set

```

for col in X_train.columns:
    if X_train[col].isnull().mean()>0:
        print(col, round(X_train[col].isnull().mean(),4))

Bare_Nuclei 0.0233

```

✓ [114] # impute missing values in X_train and X_test with respective column median in X_train

```

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)

```

✓ 0s completed at 12:55 AM

00 knn-classifier-tutorial.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```

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)

```

✓ [115] # 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

```

✓ [116] # check missing values in numerical variables in X_test

```

X_test.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

```

✓ [117] X_train.head()

✓ 0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

RAM Disk

Comment Share

+ Code + Text

[117] X_train.head()

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoli	Mitoses
293	10	4	4	6	2	10.0	2	3	1
62	9	10	10	1	10	8.0	3	3	1
485	1	1	1	3	1	3.0	1	1	1
422	4	3	3	1	2	1.0	3	3	1
332	5	2	2	2	2	1.0	2	2	1

[118] X_test.head()

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoli	Mitoses
476	4	1	2	1	2	1.0	1	1	1
531	4	2	2	1	2	1.0	2	1	1
40	6	6	6	9	6	1.0	7	8	1
432	5	1	1	1	2	1.0	2	2	1
14	8	7	5	10	7	9.0	5	5	4

Feature Scaling

[119] cols = X_train.columns

[120] from sklearn.preprocessing import StandardScaler

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

RAM Disk

Comment Share

+ Code + Text

[119] cols = X_train.columns

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)

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

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

[123] X_train.head()

	Clump_thickness	Uniformity_Cell_Size	Uniformity_Cell_Shape	Marginal_Adhesion	Single_Epithelial_Cell_Size	Bare_Nuclei	Bland_Chromatin	Normal_Nucleoli	Mitoses
0	2.028383	0.299506	0.289573	1.119077	-0.546543	1.858357	-0.577774	0.041241	-0.324258
1	1.669451	2.257680	2.304569	-0.622471	3.106879	1.297589	-0.159953	0.041241	-0.324258
2	-1.202005	-0.679581	-0.717925	0.074148	-1.003220	-0.104329	-0.995595	-0.608165	-0.324258
3	-0.125209	-0.026856	-0.046260	-0.622471	-0.546543	-0.665096	-0.159953	0.041241	-0.324258
4	0.233723	-0.353219	-0.382092	-0.274161	-0.546543	-0.665096	-0.577774	-0.283462	-0.324258

We now have X_train dataset ready to be fed into the Logistic Regression classifier. I will do it as follows.

Fit K Neighbours Classifier to the training eet

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

+ Code + Text

RAM Disk

Editing

Fit K Neighbours Classifier to the training set

```

[124] # 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)

```

Predict test-set results

```

[125] y_pred = knn.predict(X_test)

      y_pred

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

```

Check accuracy score

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

+ Code + Text

RAM Disk

Editing

2, 4, 4, 2, 2, 4, 2, 2])

Check accuracy score

```

[126] 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

      Here, y_test are the true class labels and y_pred are the predicted class labels in the test-set.

```

Compare the train-set and test-set accuracy

Now, I will compare the train-set and test-set accuracy to check for overfitting.

```

[127] y_pred_train = knn.predict(X_train)

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

      Training-set accuracy score: 0.9821

```

Check for overfitting and underfitting

```

[129] # 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

```

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

+ Code + Text

```

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

```

The training-set accuracy score is 0.9821 while the test-set accuracy to be 0.9714. These two values are quite comparable. So, there is no question of overfitting.

Rebuild kNN Classification model using different values of k

Rebuild kNN Classification model using k=5

```

[130] # 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 : {:.4f}'.format(accuracy_score(y_test, y_pred_5)))

```

```

Model accuracy score with k=5 : 0.9714

```

Rebuild kNN Classification model using k=6

0s completed at 12:55 AM

knn-classifier-tutorial.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Settings Help

+ Code + Text

```

Model accuracy score with k=5 : 0.9714

```

Rebuild kNN Classification model using k=6

```

[131] # 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 : {:.4f}'.format(accuracy_score(y_test, y_pred_6)))

```

```

Model accuracy score with k=6 : 0.9786

```

Rebuild kNN Classification model using k=7

```

[132] # 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 : {:.4f}'.format(accuracy_score(y_test, y_pred_7)))

```

```

Model accuracy score with k=7 : 0.9714

```

0s completed at 12:55 AM

```
# 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

[133] # 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

[134] # 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)
```

```
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

▼ Interpretation

Our 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 would 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.

Now, based on the above analysis we can conclude that our classification model accuracy is very good. Our model is doing a very good job in terms of predicting the class labels.

But, it does not give the underlying distribution of values. Also, it does not tell anything about the type of errors our classifier is making.

We have another tool called Confusion matrix that comes to our rescue.
```

DISCUSSION and CONCLUSION

The KNN based classification algorithm has been applied and executed successfully over a dataset.

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