Sri Sivasubramaniya Nadar College of Engineering, Chennai
(An autonomous Institution affiliated to Anna University)

Degree & Branch	B.E. Computer Science & Engineering	Semester	VI
Subject Code & Name	UCS2612 – Machine Learning Algorithms Laboratory		
Academic Year	2025–2026 (Even)	Batch	2023–2027
Due Date	**27.01.2026**		

Experiment 2: Binary Classification using Naïve Bayes and K-Nearest Neighbors

Objective

To implement Naïve Bayes and K-Nearest Neighbors (KNN) classifiers for a binary classification problem, evaluate them using multiple performance metrics, visualize model behavior, and analyze overfitting, underfitting, and bias–variance characteristics.

Dataset

A benchmark binary classification dataset containing numerical features and two class labels is used.

Dataset reference:

- Kaggle: Spambase Dataset

[23]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import time
import math
from matplotlib import rcParams
from sklearn.model_selection import train_test_split, StratifiedKFold,␣
↪GridSearchCV, RandomizedSearchCV,cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix, roc_curve, auc
rcParams['font.family']='Arial'
rcParams['font.weight']='bold'
rcParams['font.size']=15
rcParams['axes.labelweight']='bold'
rcParams['axes.titleweight']='bold'
rcParams['xtick.labelsize']=15
rcParams['ytick.labelsize']=15
```

```
from sklearn.metrics import (
accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, classification_report, roc_auc_score,
roc_curve, average_precision_score)
```

[18]:
```
#Load the dataset
df=pd.read_csv('spambase_csv_Kaggle.csv')
df
```

[18]:
```
      word_freq_make  word_freq_address  word_freq_all  word_freq_3d  \
0               0.00               0.64           0.64           0.0
1               0.21               0.28           0.50           0.0
2               0.06               0.00           0.71           0.0
3               0.00               0.00           0.00           0.0
4               0.00               0.00           0.00           0.0
...              ...                ...            ...           ...
4596            0.31               0.00           0.62           0.0
4597            0.00               0.00           0.00           0.0
4598            0.30               0.00           0.30           0.0
4599            0.96               0.00           0.00           0.0
4600            0.00               0.00           0.65           0.0

      word_freq_our  word_freq_over  word_freq_remove  word_freq_internet  \
0              0.32            0.00              0.00                0.00
1              0.14            0.28              0.21                0.07
2              1.23            0.19              0.19                0.12
3              0.63            0.00              0.31                0.63
4              0.63            0.00              0.31                0.63
...             ...             ...               ...                 ...
4596           0.00            0.31              0.00                0.00
4597           0.00            0.00              0.00                0.00
4598           0.00            0.00              0.00                0.00
4599           0.32            0.00              0.00                0.00
4600           0.00            0.00              0.00                0.00

      word_freq_order  word_freq_mail  ...  char_freq_%3B  char_freq_%28  \
0                0.00            0.00  ...          0.000          0.000
1                0.00            0.94  ...          0.000          0.132
2                0.64            0.25  ...          0.010          0.143
3                0.31            0.63  ...          0.000          0.137
4                0.31            0.63  ...          0.000          0.135
...               ...             ...  ...            ...            ...
4596             0.00            0.00  ...          0.000          0.232
4597             0.00            0.00  ...          0.000          0.000
4598             0.00            0.00  ...          0.102          0.718
4599             0.00            0.00  ...          0.000          0.057
```

```
4600            0.00            0.00  ...          0.000          0.000

      char_freq_%5B  char_freq_%21  char_freq_%24  char_freq_%23  \
0               0.0          0.778          0.000          0.000
1               0.0          0.372          0.180          0.048
2               0.0          0.276          0.184          0.010
3               0.0          0.137          0.000          0.000
4               0.0          0.135          0.000          0.000
...             ...            ...            ...            ...
4596            0.0          0.000          0.000          0.000
4597            0.0          0.353          0.000          0.000
4598            0.0          0.000          0.000          0.000
4599            0.0          0.000          0.000          0.000
4600            0.0          0.125          0.000          0.000

      capital_run_length_average  capital_run_length_longest  \
0                          3.756                          61
1                          5.114                         101
2                          9.821                         485
3                          3.537                          40
4                          3.537                          40
...                          ...                         ...
4596                       1.142                           3
4597                       1.555                           4
4598                       1.404                           6
4599                       1.147                           5
4600                       1.250                           5

      capital_run_length_total  class
0                          278      1
1                         1028      1
2                         2259      1
3                          191      1
4                          191      1
...                        ...    ...
4596                        88      0
4597                        14      0
4598                       118      0
4599                        78      0
4600                        40      0

[4601 rows x 58 columns]
```

[7]:
```
#Perform Exploratory Data Analysis (EDA)
df.describe()
```

[7]:

	word_freq_make	word_freq_address	word_freq_all	word_freq_3d	\
count	4601.000000	4601.000000	4601.000000	4601.000000	
mean	0.104553	0.213015	0.280656	0.065425	
std	0.305358	1.290575	0.504143	1.395151	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	0.420000	0.000000	
max	4.540000	14.280000	5.100000	42.810000	

	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet	\
count	4601.000000	4601.000000	4601.000000	4601.000000	
mean	0.312223	0.095901	0.114208	0.105295	
std	0.672513	0.273824	0.391441	0.401071	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	
75%	0.380000	0.000000	0.000000	0.000000	
max	10.000000	5.880000	7.270000	11.110000	

	word_freq_order	word_freq_mail	...	char_freq_%3B	char_freq_%28	\
count	4601.000000	4601.000000	...	4601.000000	4601.000000	
mean	0.090067	0.239413	...	0.038575	0.139030	
std	0.278616	0.644755	...	0.243471	0.270355	
min	0.000000	0.000000	...	0.000000	0.000000	
25%	0.000000	0.000000	...	0.000000	0.000000	
50%	0.000000	0.000000	...	0.000000	0.065000	
75%	0.000000	0.160000	...	0.000000	0.188000	
max	5.260000	18.180000	...	4.385000	9.752000	

	char_freq_%5B	char_freq_%21	char_freq_%24	char_freq_%23	\
count	4601.000000	4601.000000	4601.000000	4601.000000	
mean	0.016976	0.269071	0.075811	0.044238	
std	0.109394	0.815672	0.245882	0.429342	
min	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.315000	0.052000	0.000000	
max	4.081000	32.478000	6.003000	19.829000	

	capital_run_length_average	capital_run_length_longest	\
count	4601.000000	4601.000000	
mean	5.191515	52.172789	
std	31.729449	194.891310	
min	1.000000	1.000000	
25%	1.588000	6.000000	
50%	2.276000	15.000000	

```
75%                    3.706000                  43.000000
max                 1102.500000                9989.000000

       capital_run_length_total        class
count               4601.000000  4601.000000
mean                 283.289285     0.394045
std                  606.347851     0.488698
min                    1.000000     0.000000
25%                   35.000000     0.000000
50%                   95.000000     0.000000
75%                  266.000000     1.000000
max                15841.000000     1.000000

[8 rows x 58 columns]
```

[4]:

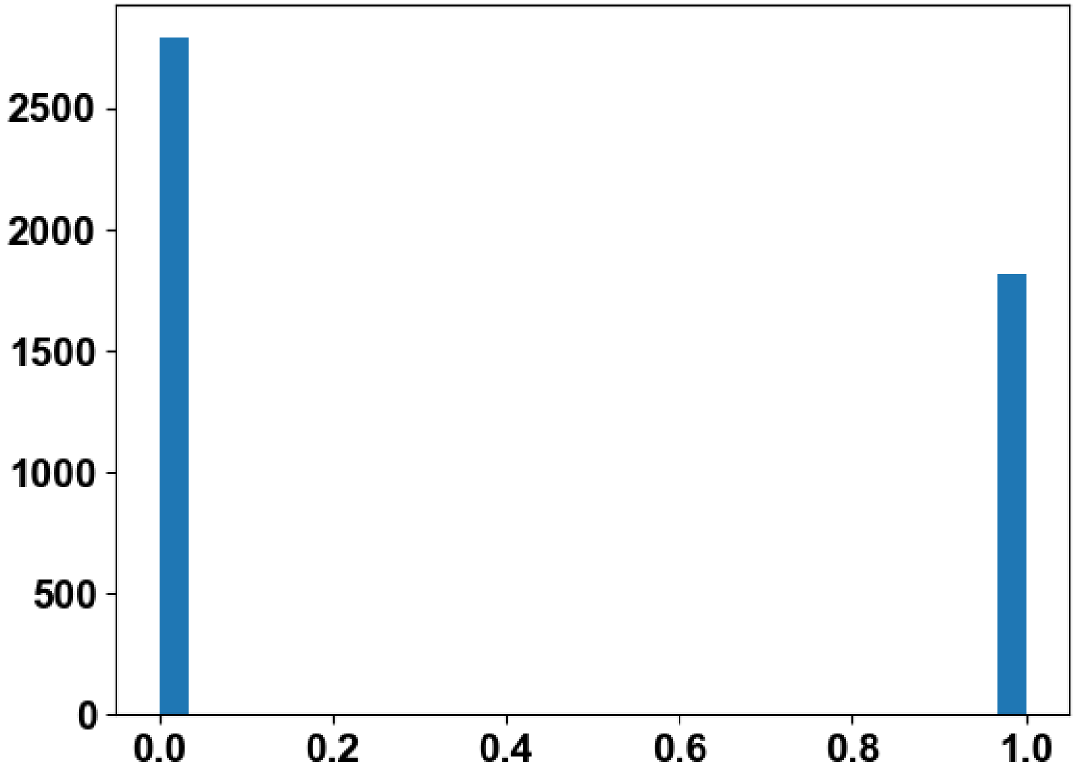
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4601 entries, 0 to 4600
Data columns (total 58 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   word_freq_make              4601 non-null   float64
 1   word_freq_address           4601 non-null   float64
 2   word_freq_all               4601 non-null   float64
 3   word_freq_3d                4601 non-null   float64
 4   word_freq_our               4601 non-null   float64
 5   word_freq_over              4601 non-null   float64
 6   word_freq_remove            4601 non-null   float64
 7   word_freq_internet          4601 non-null   float64
 8   word_freq_order             4601 non-null   float64
 9   word_freq_mail              4601 non-null   float64
 10  word_freq_receive           4601 non-null   float64
 11  word_freq_will              4601 non-null   float64
 12  word_freq_people            4601 non-null   float64
 13  word_freq_report            4601 non-null   float64
 14  word_freq_addresses         4601 non-null   float64
 15  word_freq_free              4601 non-null   float64
 16  word_freq_business          4601 non-null   float64
 17  word_freq_email             4601 non-null   float64
 18  word_freq_you               4601 non-null   float64
 19  word_freq_credit            4601 non-null   float64
 20  word_freq_your              4601 non-null   float64
 21  word_freq_font              4601 non-null   float64
 22  word_freq_000               4601 non-null   float64
 23  word_freq_money             4601 non-null   float64
 24  word_freq_hp                4601 non-null   float64
 25  word_freq_hpl               4601 non-null   float64
```

```
 26   word_freq_george            4601 non-null   float64
 27   word_freq_650               4601 non-null   float64
 28   word_freq_lab               4601 non-null   float64
 29   word_freq_labs              4601 non-null   float64
 30   word_freq_telnet            4601 non-null   float64
 31   word_freq_857               4601 non-null   float64
 32   word_freq_data              4601 non-null   float64
 33   word_freq_415               4601 non-null   float64
 34   word_freq_85                4601 non-null   float64
 35   word_freq_technology        4601 non-null   float64
 36   word_freq_1999              4601 non-null   float64
 37   word_freq_parts             4601 non-null   float64
 38   word_freq_pm                4601 non-null   float64
 39   word_freq_direct            4601 non-null   float64
 40   word_freq_cs                4601 non-null   float64
 41   word_freq_meeting           4601 non-null   float64
 42   word_freq_original          4601 non-null   float64
 43   word_freq_project           4601 non-null   float64
 44   word_freq_re                4601 non-null   float64
 45   word_freq_edu               4601 non-null   float64
 46   word_freq_table             4601 non-null   float64
 47   word_freq_conference        4601 non-null   float64
 48   char_freq_%3B               4601 non-null   float64
 49   char_freq_%28               4601 non-null   float64
 50   char_freq_%5B               4601 non-null   float64
 51   char_freq_%21               4601 non-null   float64
 52   char_freq_%24               4601 non-null   float64
 53   char_freq_%23               4601 non-null   float64
 54   capital_run_length_average  4601 non-null   float64
 55   capital_run_length_longest  4601 non-null   int64
 56   capital_run_length_total    4601 non-null   int64
 57   class                       4601 non-null   int64
dtypes: float64(55), int64(3)
memory usage: 2.0 MB
```

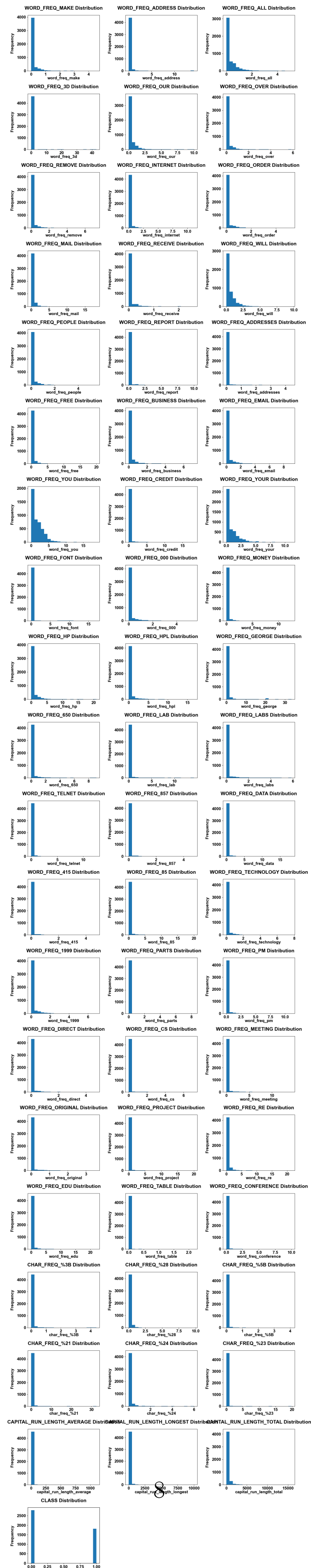
[24]:

```
# isualize class distribution and feature behavior
plt.hist(df["class"],bins=30)
plt.savefig("class_distribution.png", dpi=300, bbox_inches="tight")
```



Histogram

[13]:
```
bxwidth=1
rows=math.ceil(len(df.columns)/3)
fig,axes=plt.subplots(rows,3,figsize=(15,4*rows))
axes=axes.flatten()
columns=df.columns
for i,(ax,col) in enumerate(zip(axes,columns)):
  ax.hist(df[col],bins=20)
  ax.set_title(f"{col.upper()} Distribution",pad=20)
  ax.set_xlabel(col,labelpad=0)
  ax.set_ylabel("Frequency",labelpad=10)
for spine in ax.spines.values():
  spine.set_linewidth(bxwidth)
for j in range(i+1,len(axes)):
  fig.delaxes(axes[j])
plt.subplots_adjust(hspace=1.5,wspace=1.3)
plt.tight_layout()
plt.savefig("histogram.png", dpi=300, bbox_inches="tight")
plt.show()
```


WORD_FREQ_MAKE Distribution
word_freq_make
WORD_FREQ_ADDRESS Distribution
word_freq_address
WORD_FREQ_ALL Distribution
word_freq_all
WORD_FREQ_3D Distribution
word_freq_3d
WORD_FREQ_OUR Distribution
word_freq_our
WORD_FREQ_OVER Distribution
word_freq_over
WORD_FREQ_REMOVE Distribution
word_freq_remove
WORD_FREQ_INTERNET Distribution
word_freq_internet
WORD_FREQ_ORDER Distribution
word_freq_order
WORD_FREQ_MAIL Distribution
word_freq_mail
WORD_FREQ_RECEIVE Distribution
word_freq_receive
WORD_FREQ_WILL Distribution
word_freq_will
WORD_FREQ_PEOPLE Distribution
word_freq_people
WORD_FREQ_REPORT Distribution
word_freq_report
WORD_FREQ_ADDRESSES Distribution
word_freq_addresses
WORD_FREQ_FREE Distribution
word_freq_free
WORD_FREQ_BUSINESS Distribution
word_freq_business
WORD_FREQ_EMAIL Distribution
word_freq_email
WORD_FREQ_YOU Distribution
word_freq_you
WORD_FREQ_CREDIT Distribution
word_freq_credit
WORD_FREQ_YOUR Distribution
word_freq_your
WORD_FREQ_FONT Distribution
word_freq_font
WORD_FREQ_000 Distribution
word_freq_000
WORD_FREQ_MONEY Distribution
word_freq_money
WORD_FREQ_HP Distribution
word_freq_hp
WORD_FREQ_HPL Distribution
word_freq_hpl
WORD_FREQ_GEORGE Distribution
word_freq_george
WORD_FREQ_650 Distribution
word_freq_650
WORD_FREQ_LAB Distribution
word_freq_lab
WORD_FREQ_LABS Distribution
word_freq_labs
WORD_FREQ_TELNET Distribution
word_freq_telnet
WORD_FREQ_857 Distribution
word_freq_857
WORD_FREQ_DATA Distribution
word_freq_data
WORD_FREQ_415 Distribution
word_freq_415
WORD_FREQ_85 Distribution
word_freq_85
WORD_FREQ_TECHNOLOGY Distribution
word_freq_technology
WORD_FREQ_1999 Distribution
word_freq_1999
WORD_FREQ_PARTS Distribution
word_freq_parts
WORD_FREQ_PM Distribution
word_freq_pm
WORD_FREQ_DIRECT Distribution
word_freq_direct
WORD_FREQ_CS Distribution
word_freq_cs
WORD_FREQ_MEETING Distribution
word_freq_meeting
WORD_FREQ_ORIGINAL Distribution
word_freq_original
WORD_FREQ_PROJECT Distribution
word_freq_project
WORD_FREQ_RE Distribution
word_freq_re
WORD_FREQ_EDU Distribution
word_freq_edu
WORD_FREQ_TABLE Distribution
word_freq_table
WORD_FREQ_CONFERENCE Distribution
word_freq_conference
CHAR_FREQ_%3B Distribution
char_freq_%3B
CHAR_FREQ_%28 Distribution
char_freq_%28
CHAR_FREQ_%5B Distribution
char_freq_%5B
CHAR_FREQ_%21 Distribution
char_freq_%21
CHAR_FREQ_%24 Distribution
char_freq_%24
CHAR_FREQ_%23 Distribution
char_freq_%23
CAPITAL_RUN_LENGTH_AVERAGE Distribution
capital_run_length_average
CAPITAL_RUN_LENGTH_LONGEST Distribution
capital_run_length_longest
CAPITAL_RUN_LENGTH_TOTAL Distribution
capital_run_length_total
CLASS Distribution
class
Frequency

Train-Test Split

[19]:
```
X=df.iloc[:,:-1]
y=df.iloc[:,-1]
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.
 ↪2,stratify=y,random_state=42)
print(X_train,X_test,y_train,y_test)
```

```
      word_freq_make  word_freq_address  word_freq_all  word_freq_3d  \
2940            0.05               0.00           0.45           0.0
1303            0.17               0.26           1.21           0.0
3468            0.00               0.00           0.00           0.0
3181            0.00               0.00           0.00           0.0
794             0.00               0.56           0.00           0.0
...              ...                ...            ...           ...
1861            0.00               0.00           4.00           0.0
2366            0.00               0.00           0.00           0.0
330             0.00               0.00           1.53           0.0
536             0.00               0.00           0.00           0.0
3114            0.00               0.00           0.00           0.0

      word_freq_our  word_freq_over  word_freq_remove  word_freq_internet  \
2940           0.15             0.1              0.00                0.00
1303           0.43             0.6              0.43                0.26
3468           0.00             0.0              0.00                0.00
3181           0.00             0.0              0.00                0.00
794            0.56             0.0              0.00                0.00
...             ...             ...               ...                 ...
1861           0.00             0.0              0.00                0.00
2366           4.16             0.0              0.00                0.00
330            0.00             0.0              0.00                0.00
536            0.00             0.0              0.00                0.00
3114           0.00             0.0              0.00                0.00

      word_freq_order  word_freq_mail  ...  word_freq_conference  \
2940             0.55            0.00  ...                   0.0
1303             0.69            0.52  ...                   0.0
3468             0.00            0.00  ...                   0.0
3181             0.00            0.00  ...                   0.0
794              1.01            0.56  ...                   0.0
...               ...             ...  ...                   ...
1861             0.00            0.00  ...                   0.0
2366             0.00            0.00  ...                   0.0
330              0.00            0.00  ...                   0.0
536              0.00            0.00  ...                   0.0
```

```
3114             0.00            0.00  ...                 0.0

      char_freq_%3B  char_freq_%28  char_freq_%5B  char_freq_%21  \
2940          0.203          0.195           0.05          0.000
1303          0.000          0.108           0.00          0.271
3468          0.000          0.000           0.00          0.153
3181          0.000          0.000           0.00          0.000
794           0.000          0.186           0.00          0.056
...             ...            ...            ...            ...
1861          0.000          0.000           0.00          0.613
2366          0.000          0.689           0.00          0.689
330           0.000          0.000           0.00          1.434
536           0.000          0.407           0.00          0.203
3114          0.000          0.484           0.00          0.484

      char_freq_%24  char_freq_%23  capital_run_length_average  \
2940          0.014          0.000                       2.880
1303          0.243          0.013                       6.395
3468          0.000          0.000                       1.933
3181          0.000          0.000                       4.333
794           0.056          0.000                       2.153
...             ...            ...                         ...
1861          0.000          0.000                       1.000
2366          0.000          0.000                       1.300
330           0.000          0.000                       7.055
536           0.610          0.000                       4.133
3114          0.000          0.000                       2.500

      capital_run_length_longest  capital_run_length_total
2940                          45                      1080
1303                         583                      1375
3468                           7                        58
3181                          20                        26
794                           53                       532
...                          ...                       ...
1861                           1                        14
2366                           4                        13
330                           75                       127
536                           17                        62
3114                          15                        65

[3680 rows x 57 columns]       word_freq_make  word_freq_address  word_freq_all
word_freq_3d  \
1472            0.00               0.00           0.00           0.0
258             0.00               0.00           0.33           0.0
3564            0.00               0.00           0.00           0.0
65              0.66               0.00           0.66           0.0
4303            0.00               0.00           0.00           0.0
```

```
...             ...                ...          ...          ...
1405           0.00               0.00         0.35          0.0
2312           0.00               2.59         1.29          0.0
2804           0.00               0.00         0.00          0.0
2047           0.00               0.00         0.00          0.0
2597           0.00               0.00         0.00          0.0

      word_freq_our  word_freq_over  word_freq_remove  word_freq_internet  \
1472           0.00            0.00              0.00                0.00
258            0.99            0.99              0.33                0.33
3564           0.00            0.00              0.00                0.00
65             0.00            0.00              0.00                0.00
4303           0.00            0.00              0.00                0.00
...             ...             ...               ...                 ...
1405           0.00            0.70              0.35                0.35
2312           1.29            0.00              0.00                0.00
2804           0.00            0.00              0.00                0.00
2047           0.34            0.00              0.00                0.00
2597           0.00            0.00              0.00                0.00

      word_freq_order  word_freq_mail  ...  word_freq_conference  \
1472              0.0            0.00  ...                  0.00
258               0.0            0.00  ...                  0.00
3564              0.0            0.00  ...                  0.00
65                0.0            0.66  ...                  0.00
4303              0.0            2.77  ...                  0.00
...               ...             ...  ...                   ...
1405              0.0            0.00  ...                  0.00
2312              0.0            0.00  ...                  0.00
2804              0.0            0.00  ...                  0.00
2047              0.0            0.00  ...                  0.34
2597              0.0            0.00  ...                  0.00

      char_freq_%3B  char_freq_%28  char_freq_%5B  char_freq_%21  \
1472          0.144          0.000          0.000          3.907
258           0.000          0.108          0.000          0.000
3564          0.000          0.000          0.000          0.000
65            0.000          0.000          0.000          2.205
4303          0.000          0.000          0.000          0.438
...             ...            ...            ...            ...
1405          0.000          0.061          0.000          0.061
2312          0.000          0.000          0.000          0.000
2804          0.000          0.000          0.000          0.000
2047          0.088          0.132          0.000          0.000
2597          0.142          0.000          0.142          0.000

      char_freq_%24  char_freq_%23  capital_run_length_average  \
1472          0.000          0.000                      13.928
```

```
258          0.162      0.054                         2.195
3564         0.000      0.000                         1.214
65           0.000      0.000                         3.184
4303         0.000      0.000                         1.214
...            ...        ...                           ...
1405         0.000      0.122                         2.302
2312         0.000      0.000                         1.000
2804         0.000      0.000                         2.000
2047         0.000      0.000                         1.250
2597         0.000      0.000                         1.717

      capital_run_length_longest  capital_run_length_total
1472                          70                       195
258                           50                       202
3564                           4                        17
65                            34                       121
4303                           3                        17
...                          ...                       ...
1405                          21                        99
2312                           1                        13
2804                           4                         6
2047                           7                        85
2597                          12                        67

[921 rows x 57 columns] 2940    0
1303    1
3468    0
3181    0
794     1
       ..
1861    0
2366    0
330     1
536     1
3114    0
Name: class, Length: 3680, dtype: int64 1472    1
258     1
3564    0
65      1
4303    0
       ..
1405    1
2312    0
2804    0
2047    0
2597    0
Name: class, Length: 921, dtype: int64
```

Z-score or Standard Scaling

[21]:
```python
scaler=StandardScaler()
X_train_scaled=scaler.fit_transform(X_train)
X_test_scaled=scaler.transform(X_test)
print(X_train,X_test,y_train,y_test)
```

```
      word_freq_make  word_freq_address  word_freq_all  word_freq_3d  \
2940            0.05               0.00           0.45           0.0
1303            0.17               0.26           1.21           0.0
3468            0.00               0.00           0.00           0.0
3181            0.00               0.00           0.00           0.0
794             0.00               0.56           0.00           0.0
...              ...                ...            ...           ...
1861            0.00               0.00           4.00           0.0
2366            0.00               0.00           0.00           0.0
330             0.00               0.00           1.53           0.0
536             0.00               0.00           0.00           0.0
3114            0.00               0.00           0.00           0.0

      word_freq_our  word_freq_over  word_freq_remove  word_freq_internet  \
2940           0.15             0.1              0.00                0.00
1303           0.43             0.6              0.43                0.26
3468           0.00             0.0              0.00                0.00
3181           0.00             0.0              0.00                0.00
794            0.56             0.0              0.00                0.00
...             ...             ...               ...                 ...
1861           0.00             0.0              0.00                0.00
2366           4.16             0.0              0.00                0.00
330            0.00             0.0              0.00                0.00
536            0.00             0.0              0.00                0.00
3114           0.00             0.0              0.00                0.00

      word_freq_order  word_freq_mail  ...  word_freq_conference  \
2940             0.55            0.00  ...                   0.0
1303             0.69            0.52  ...                   0.0
3468             0.00            0.00  ...                   0.0
3181             0.00            0.00  ...                   0.0
794              1.01            0.56  ...                   0.0
...               ...             ...  ...                   ...
1861             0.00            0.00  ...                   0.0
2366             0.00            0.00  ...                   0.0
330              0.00            0.00  ...                   0.0
536              0.00            0.00  ...                   0.0
3114             0.00            0.00  ...                   0.0

      char_freq_%3B  char_freq_%28  char_freq_%5B  char_freq_%21  \
2940          0.203          0.195           0.05          0.000
```

```
1303          0.000          0.108          0.00          0.271
3468          0.000          0.000          0.00          0.153
3181          0.000          0.000          0.00          0.000
794           0.000          0.186          0.00          0.056
...             ...            ...           ...            ...
1861          0.000          0.000          0.00          0.613
2366          0.000          0.689          0.00          0.689
330           0.000          0.000          0.00          1.434
536           0.000          0.407          0.00          0.203
3114          0.000          0.484          0.00          0.484

      char_freq_%24  char_freq_%23  capital_run_length_average  \
2940          0.014          0.000                       2.880
1303          0.243          0.013                       6.395
3468          0.000          0.000                       1.933
3181          0.000          0.000                       4.333
794           0.056          0.000                       2.153
...             ...            ...                         ...
1861          0.000          0.000                       1.000
2366          0.000          0.000                       1.300
330           0.000          0.000                       7.055
536           0.610          0.000                       4.133
3114          0.000          0.000                       2.500

      capital_run_length_longest  capital_run_length_total
2940                          45                      1080
1303                         583                      1375
3468                           7                        58
3181                          20                        26
794                           53                       532
...                          ...                       ...
1861                           1                        14
2366                           4                        13
330                           75                       127
536                           17                        62
3114                          15                        65

[3680 rows x 57 columns]       word_freq_make  word_freq_address  word_freq_all
word_freq_3d  \
1472            0.00               0.00           0.00           0.0
258             0.00               0.00           0.33           0.0
3564            0.00               0.00           0.00           0.0
65              0.66               0.00           0.66           0.0
4303            0.00               0.00           0.00           0.0
...              ...                ...            ...           ...
1405            0.00               0.00           0.35           0.0
2312            0.00               2.59           1.29           0.0
2804            0.00               0.00           0.00           0.0
```

2047	0.00	0.00	0.00	0.0
2597	0.00	0.00	0.00	0.0

	word_freq_our	word_freq_over	word_freq_remove	word_freq_internet \
1472	0.00	0.00	0.00	0.00
258	0.99	0.99	0.33	0.33
3564	0.00	0.00	0.00	0.00
65	0.00	0.00	0.00	0.00
4303	0.00	0.00	0.00	0.00
...	...	...	...	...
1405	0.00	0.70	0.35	0.35
2312	1.29	0.00	0.00	0.00
2804	0.00	0.00	0.00	0.00
2047	0.34	0.00	0.00	0.00
2597	0.00	0.00	0.00	0.00

	word_freq_order	word_freq_mail	...	word_freq_conference \
1472	0.0	0.00	...	0.00
258	0.0	0.00	...	0.00
3564	0.0	0.00	...	0.00
65	0.0	0.66	...	0.00
4303	0.0	2.77	...	0.00
...	...	...	...	...
1405	0.0	0.00	...	0.00
2312	0.0	0.00	...	0.00
2804	0.0	0.00	...	0.00
2047	0.0	0.00	...	0.34
2597	0.0	0.00	...	0.00

	char_freq_%3B	char_freq_%28	char_freq_%5B	char_freq_%21 \
1472	0.144	0.000	0.000	3.907
258	0.000	0.108	0.000	0.000
3564	0.000	0.000	0.000	0.000
65	0.000	0.000	0.000	2.205
4303	0.000	0.000	0.000	0.438
...	...	...	...	...
1405	0.000	0.061	0.000	0.061
2312	0.000	0.000	0.000	0.000
2804	0.000	0.000	0.000	0.000
2047	0.088	0.132	0.000	0.000
2597	0.142	0.000	0.142	0.000

	char_freq_%24	char_freq_%23	capital_run_length_average \
1472	0.000	0.000	13.928
258	0.162	0.054	2.195
3564	0.000	0.000	1.214
65	0.000	0.000	3.184
4303	0.000	0.000	1.214

```
...           ...         ...                    ...
1405          0.000       0.122                  2.302
2312          0.000       0.000                  1.000
2804          0.000       0.000                  2.000
2047          0.000       0.000                  1.250
2597          0.000       0.000                  1.717

      capital_run_length_longest  capital_run_length_total
1472                          70                       195
258                           50                       202
3564                           4                        17
65                            34                       121
4303                           3                        17
...                          ...                       ...
1405                          21                        99
2312                           1                        13
2804                           4                         6
2047                           7                        85
2597                          12                        67

[921 rows x 57 columns] 2940    0
1303    1
3468    0
3181    0
794     1
       ..
1861    0
2366    0
330     1
536     1
3114    0
Name: class, Length: 3680, dtype: int64 1472    1
258     1
3564    0
65      1
4303    0
       ..
1405    1
2312    0
2804    0
2047    0
2597    0
Name: class, Length: 921, dtype: int64
```

Naive Bayes GaussianNB

[27]:
```
start=time.time()
gnb=GaussianNB()
```

```
gnb.fit(X_train_scaled,y_train)
ttgnb=time.time()-start
y_pred=gnb.predict(X_test_scaled)
print("The training time for Gaussian Naive Bayes is: ",ttgnb)
```

```
The training time for Gaussian Naive Bayes is:  0.00466465950012207
```

[39]:
```
start=time.time()
y_pred=gnb.predict(X_test_scaled)
ptgnb=time.time()-start
print("The Prediction time for Gaussian Naive Bayes is: ",ptgnb)
```

```
The Prediction time for Gaussian Naive Bayes is:  0.0009808540344238281
```

Accuracy

[30]:
```
accuracy_score(y_test,y_pred)
```

[30]:
```
0.8327904451682954
```

Precision

[31]:
```
precision_score(y_test,y_pred)
```

[31]:
```
0.7145790554414785
```

Recall

[32]:
```
recall_score(y_test,y_pred)
```

[32]:
```
0.9586776859504132
```

F1 Score

[33]:
```
f1_score(y_test,y_pred)
```

[33]:
```
0.8188235294117647
```

[34]:
```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[419 139]
 [ 15 348]]
```

Specificity and False Positive Rate

[36]:
```
TN, FP, FN, TP = cm.ravel()
specificity = TN / (TN + FP)
false_positive_rate = FP / (FP + TN)

print("Specificity:", specificity)
print("False Positive Rate:", false_positive_rate)
```

```
Specificity: 0.7508960573476703
False Positive Rate: 0.24910394265232974
```

[37]:
```python
print("Overall Report of model \n\n",classification_report(y_test,y_pred))
```

```
Overall Report of model

              precision    recall  f1-score   support

           0       0.97      0.75      0.84       558
           1       0.71      0.96      0.82       363

    accuracy                           0.83       921
   macro avg       0.84      0.85      0.83       921
weighted avg       0.87      0.83      0.83       921
```

MultinomialNB

[40]:
```python
start=time.time()
mnb=MultinomialNB()
mnb.fit(X_train,y_train)
ttmnb=time.time()-start


print("The training time for Multinomial Naive Bayes is: ",ttmnb)
```

```
The training time for Multinomial Naive Bayes is:  0.009273767471313477
```

[41]:
```python
start=time.time()
y_pred=mnb.predict(X_test)
ptmnb=time.time()-start
print("The Prediction time for Multinomial Naive Bayes is: ",ptmnb)
```

```
The Prediction time for Multinomial Naive Bayes is:  0.009157180786132812
```

Accuracy

[42]:
```python
accuracy_score(y_test,y_pred)
```

[42]: 0.7763300760043431

Precision Score

[43]:
```python
precision_score(y_test,y_pred)
```

[43]: 0.7198879551820728

Recall-score

[44]:
```python
recall_score(y_test,y_pred)
```

[44]: 0.7079889807162535

F1-score

[45]:
```
f1_score(y_test,y_pred)
```

[45]: 0.7138888888888889

[46]:
```
print("Confusion matrix is \n",confusion_matrix(y_test,y_pred))
```

```
Confusion matrix is
 [[458 100]
 [106 257]]
```

[48]:
```
TN, FP, FN, TP = cm.ravel()
specificity = TN / (TN + FP)
false_positive_rate = FP / (FP + TN)

print("Specificity:", specificity)
print("False Positive Rate:", false_positive_rate)
```

```
Specificity: 0.7508960573476703
False Positive Rate: 0.24910394265232974
```

[47]:
```
print("Overall Report of model \n\n",classification_report(y_test,y_pred))
```

```
Overall Report of model

               precision    recall  f1-score   support

           0       0.81      0.82      0.82       558
           1       0.72      0.71      0.71       363

    accuracy                           0.78       921
   macro avg       0.77      0.76      0.77       921
weighted avg       0.78      0.78      0.78       921
```

BernoulliNB

[49]:
```
start=time.time()
bnb=BernoulliNB()
bnb.fit(X_train,y_train)
ttbnb=time.time()-start
print("The training time for Bernoulli Naive Bayes is: ",ttbnb)
```

```
The training time for Bernoulli Naive Bayes is:  0.025691270828247070
```

[50]:
```
start=time.time()
y_pred=bnb.predict(X_test)
```

```
ptbnb=time.time()-start

print("The prediction time for Bernoulli Naive Bayes is: ",ptbnb)
```

```
The prediction time for Bernoulli Naive Bayes is:  0.003880739212036133
```

Accuracy

[51]:
```
accuracy_score(y_test,y_pred)
```

[51]:
```
0.8762214983713354
```

Precision

[52]:
```
precision_score(y_test,y_pred)
```

[52]:
```
0.8716417910447761
```

Recall Score

[53]:
```
recall_score(y_test,y_pred)
```

[53]:
```
0.8044077134986226
```

F1-score

[54]:
```
f1_score(y_test,y_pred)
```

[54]:
```
0.836676217765043
```

[56]:
```
print("Confusion matrix is \n",confusion_matrix(y_test,y_pred))
```

```
Confusion matrix is
 [[515  43]
 [ 71 292]]
```

[57]:
```
TN, FP, FN, TP = cm.ravel()
specificity = TN / (TN + FP)
false_positive_rate = FP / (FP + TN)

print("Specificity:", specificity)
print("False Positive Rate:", false_positive_rate)
```

```
Specificity: 0.7508960573476703
False Positive Rate: 0.24910394265232974
```

[58]:
```
print("Overall Report of model \n\n",classification_report(y_test,y_pred))
```

```
Overall Report of model

               precision    recall  f1-score   support
```

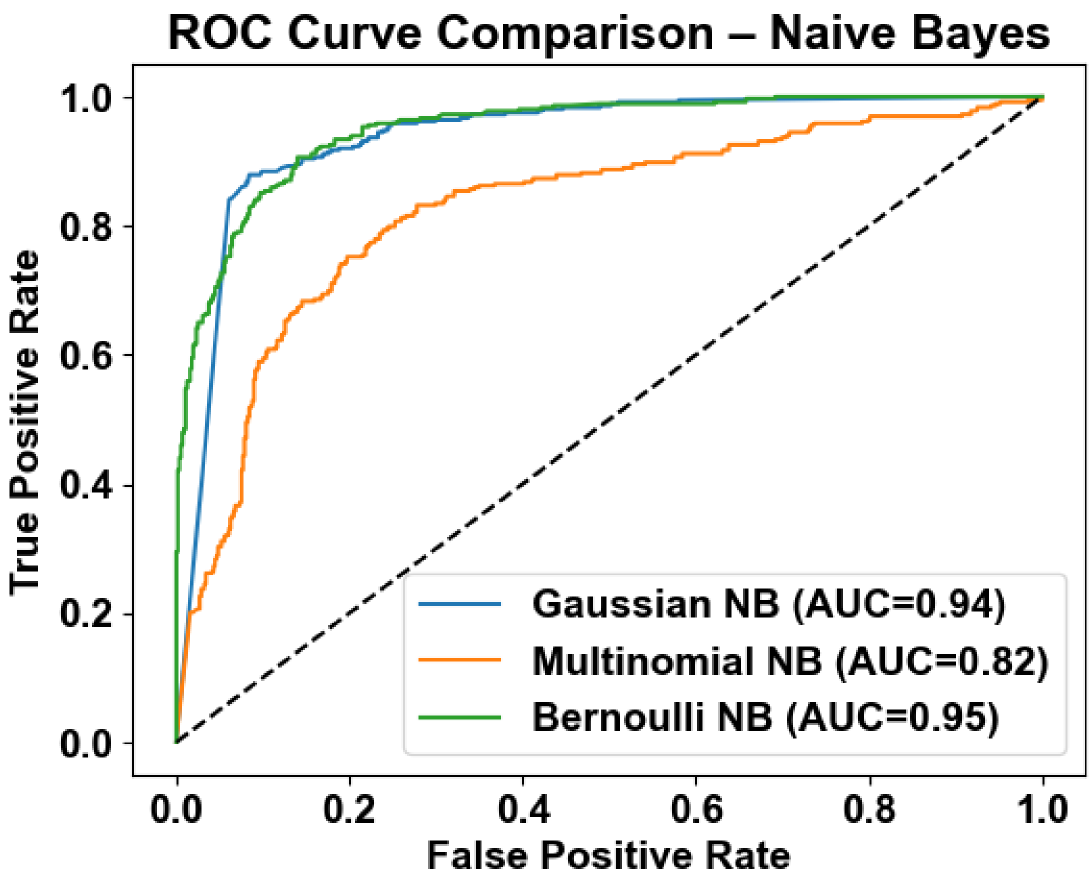
```
              0       0.88      0.92      0.90       558
              1       0.87      0.80      0.84       363

       accuracy                           0.88       921
      macro avg       0.88      0.86      0.87       921
   weighted avg       0.88      0.88      0.88       921
```

ROC curve

[59]:
```
y_prob_gnb=gnb.predict_proba(X_test_scaled)[:,1]
y_prob_mnb=mnb.predict_proba(X_test)[:,1]
y_prob_bnb=bnb.predict_proba(X_test)[:,1]
```

[60]:
```
fpr_gnb,tpr_gnb,_=roc_curve(y_test,y_prob_gnb)
fpr_mnb,tpr_mnb,_=roc_curve(y_test,y_prob_mnb)
fpr_bnb,tpr_bnb,_=roc_curve(y_test,y_prob_bnb)
auc_gnb=auc(fpr_gnb,tpr_gnb)
auc_mnb=auc(fpr_mnb,tpr_mnb)
auc_bnb=auc(fpr_bnb,tpr_bnb)
plt.plot(fpr_gnb,tpr_gnb,label=f'Gaussian NB (AUC={auc_gnb:.2f})')
plt.plot(fpr_mnb,tpr_mnb,label=f'Multinomial NB (AUC={auc_mnb:.2f})')
plt.plot(fpr_bnb,tpr_bnb,label=f'Bernoulli NB (AUC={auc_bnb:.2f})')
plt.plot([0,1],[0,1],'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve Comparison - Naive Bayes")
plt.legend()
plt.savefig("roc_curve.png", dpi=300, bbox_inches="tight")
plt.show()
```



K-Nearest Neighbour

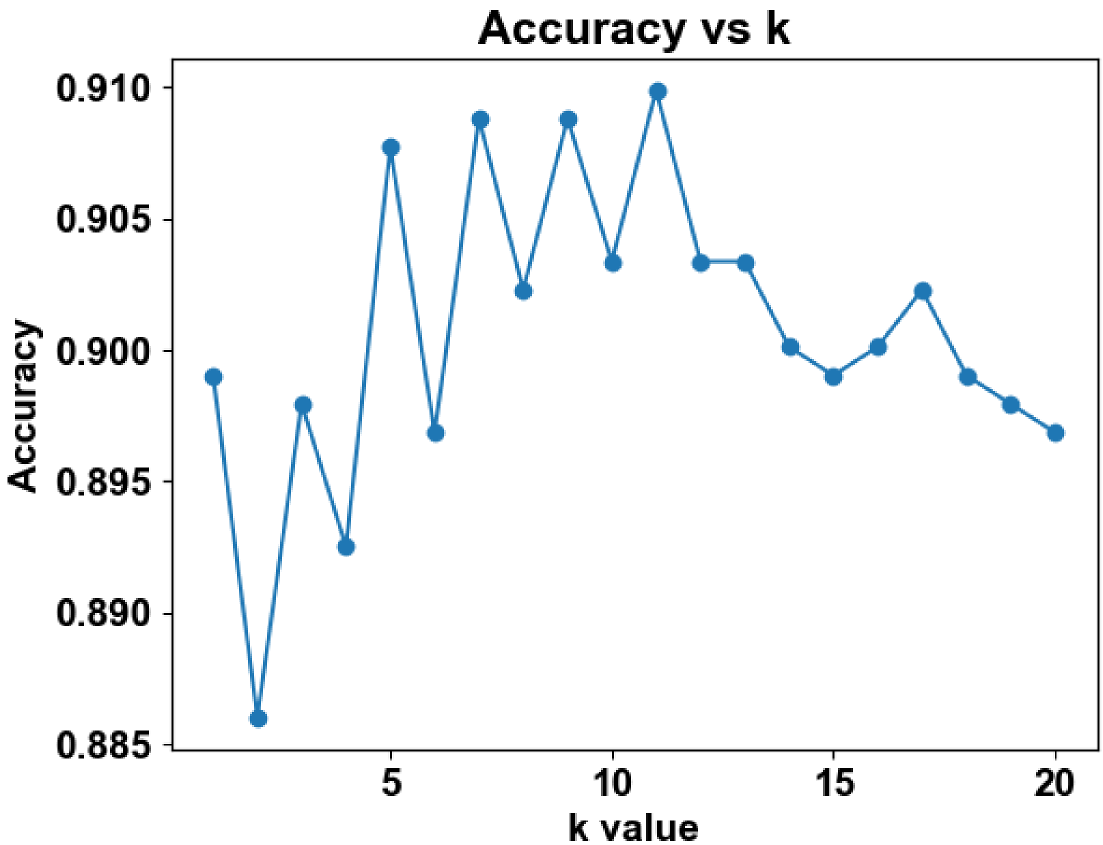
Basic Model

[61]:
```
knn = KNeighborsClassifier()
knn.fit(X_train_scaled, y_train)
y_pred_knn = knn.predict(X_test_scaled)
```

Statistical significance test Anova-N

[76]:
```
k_values = range(1, 21)
accuracies = []
for k in k_values:
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(X_train_scaled, y_train)
  accuracies.append(knn.score(X_test_scaled, y_test))
plt.plot(k_values, accuracies, marker='o')
plt.xlabel('k value')
plt.ylabel('Accuracy')
plt.title('Accuracy vs k')
```

```
plt.savefig("accuracy_vs_k.png", dpi=300, bbox_inches="tight")
plt.show()
```



Stratified K-Fold

[77]:
```
skf=StratifiedKFold(n_splits=5,shuffle=True,random_state=42)
cv_scores_base=cross_val_score(
    knn,
    X_train_scaled,
    y_train,
    cv=skf,
    scoring='accuracy'
)
print("Base KNN CV Accuracy:",cv_scores_base.mean())
```

```
Base KNN CV Accuracy: 0.8932065217391303
```

Grid Search

[78]:
```
param_grid={
    'n_neighbors':list(range(1,31,2)),
```

```python
    'weights':['uniform','distance'],
    'metric':['euclidean','manhattan']
}
grid=GridSearchCV(
    knn,
    param_grid,
    cv=skf,
    scoring='accuracy',
    n_jobs=-1
)
grid.fit(X_train_scaled,y_train)
print("Grid Best Params:",grid.best_params_)
print("Grid Best CV Accuracy:",grid.best_score_)
```

```
Grid Best Params: {'metric': 'manhattan', 'n_neighbors': 9, 'weights':
'distance'}
Grid Best CV Accuracy: 0.9252717391304348
```

Randomized Search

[79]:
```python
from scipy.stats import randint
param_dist={
    'n_neighbors':randint(1,30),
    'weights':['uniform','distance'],
    'metric':['euclidean','manhattan']
}
rand=RandomizedSearchCV(
    knn,
    param_distributions=param_dist,
    n_iter=15,
    cv=skf,
    scoring='accuracy',
    random_state=42,
    n_jobs=-1
)
rand.fit(X_train_scaled,y_train)
print("Random Best Params:",rand.best_params_)
print("Random Best CV Accuracy:",rand.best_score_)
```

```
Random Best Params: {'metric': 'manhattan', 'n_neighbors': 6, 'weights':
'distance'}
Random Best CV Accuracy: 0.9241847826086957
```

Final KNN Model

[80]:
```python
best_params=grid.best_params_
knn_final=KNeighborsClassifier(
    n_neighbors=best_params['n_neighbors'],
    weights=best_params['weights'],
```

```
    metric=best_params['metric']
)
knn_final.fit(X_train_scaled,y_train)
y_pred_final=knn_final.predict(X_test_scaled)
```

Metrics

[81]:
```
from sklearn.metrics import confusion_matrix,roc_curve,auc
def compute_metrics(y_true,y_pred):
    cm=confusion_matrix(y_true,y_pred)
    tn,fp,fn,tp=cm.ravel()
    accuracy=(tp+tn)/(tp+tn+fp+fn)
    precision=tp/(tp+fp)
    recall=tp/(tp+fn)
    f1=2*precision*recall/(precision+recall)
    specificity=tn/(tn+fp)
    fpr=fp/(fp+tn)
    return accuracy,precision,recall,f1,specificity,fpr,cm
```

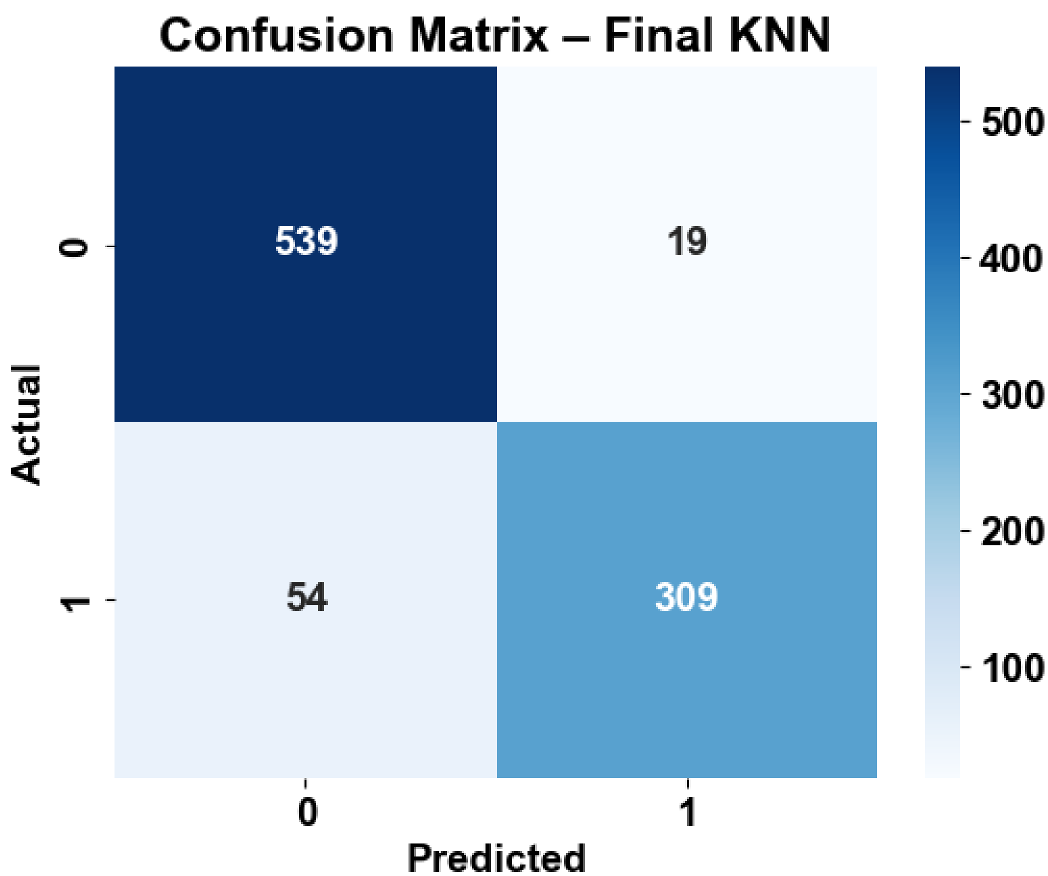
[82]:
```
start=time.time()
knn_final.fit(X_train_scaled,y_train)
train_time=time.time()-start

start=time.time()
y_pred_knn=knn_final.predict(X_test_scaled)
pred_time=time.time()-start
acc,prec,rec,f1,spec,fpr,cm=compute_metrics(y_test,y_pred_knn)
print("Final KNN Metrics")
print("Accuracy:",acc)
print("Precision:",prec)
print("Recall:",rec)
print("F1 Score:",f1)
print("Specificity:",spec)
print("False Positive Rate:",fpr)
print("Training Time:",train_time)
print("Prediction Time:",pred_time)
```

```
Final KNN Metrics
Accuracy: 0.9207383279044516
Precision: 0.9420731707317073
Recall: 0.8512396694214877
F1 Score: 0.894356005788712
Specificity: 0.9659498207885304
False Positive Rate: 0.034050179211469536
Training Time: 0.004589557647705078
Prediction Time: 0.09273862838745117
```

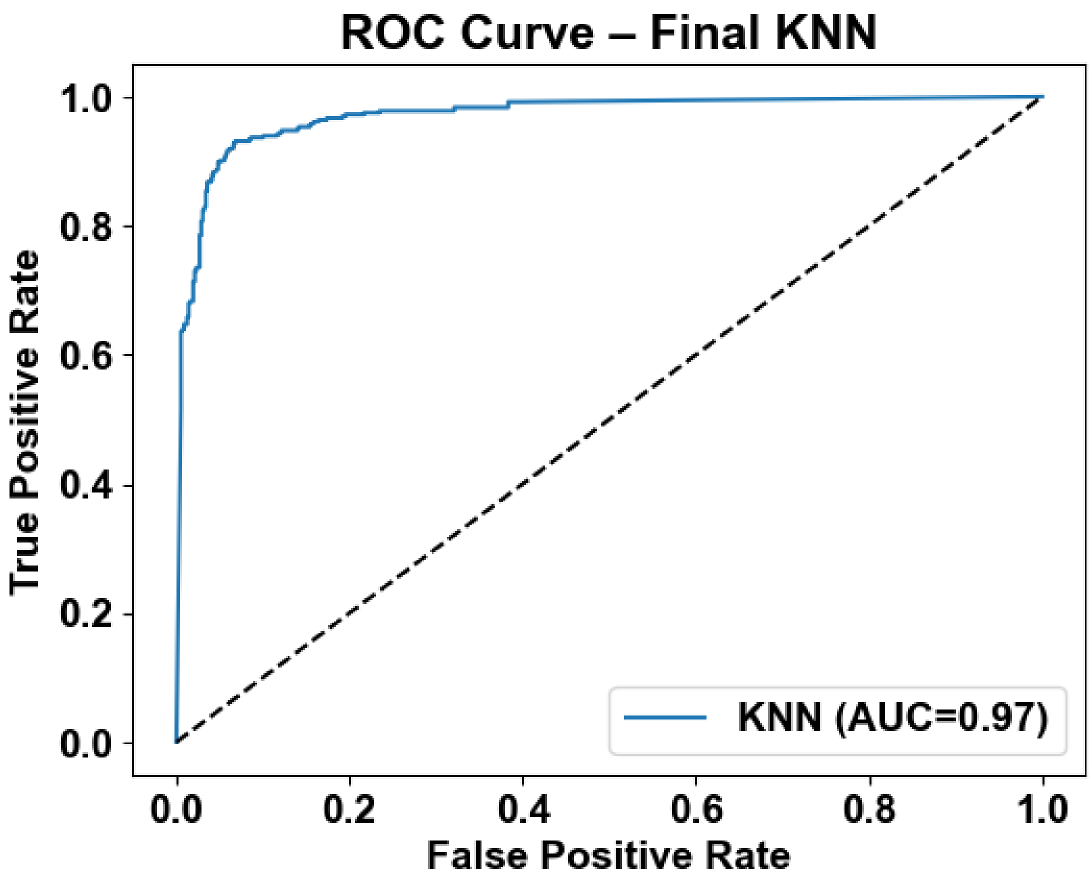
Confustion Matrix

[83]:
```
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.title("Confusion Matrix - Final KNN")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



ROC Curve for KNN

[85]:
```
y_prob_knn=knn_final.predict_proba(X_test_scaled)[:,1]
fpr_knn,tpr_knn,_=roc_curve(y_test,y_prob_knn)
auc_knn=auc(fpr_knn,tpr_knn)
plt.plot(fpr_knn,tpr_knn,label=f'KNN (AUC={auc_knn:.2f})')
plt.plot([0,1],[0,1],'k--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve - Final KNN")
plt.legend()
plt.savefig("roc_curve_finalKNN.png", dpi=300, bbox_inches="tight")
plt.show()
```



[86]:
```
best_params=grid.best_params_
optimal_k=best_params['n_neighbors']
```

Evaluating all models using multiple metrics

KDTree

[87]:
```
start=time.time()
knn_kd=KNeighborsClassifier(
    n_neighbors=optimal_k,
    weights=best_params['weights'],
    metric=best_params['metric'],
    algorithm='kd_tree'
)
knn_kd.fit(X_train_scaled,y_train)
train_time_kd=time.time()-start
start=time.time()
y_pred_kd=knn_kd.predict(X_test_scaled)
pred_time_kd=time.time()-start
```

[88]:
```python
acc_kd,prec_kd,rec_kd,f1_kd,sp,fpr,cm=compute_metrics(y_test,y_pred_kd)
print("Final KDtree Metrics")
print("Accuracy:",acc_kd)
print("Precision:",prec_kd)
print("Recall:",rec_kd)
print("F1 Score:",f1_kd)
print("Specificity:",sp)
print("False Positive Rate:",fpr)
print("Training Time:",train_time)
print("Prediction Time:",pred_time)
```

```
Final KDtree Metrics
Accuracy: 0.9207383279044516
Precision: 0.9420731707317073
Recall: 0.8512396694214877
F1 Score: 0.894356005788712
Specificity: 0.9659498207885304
False Positive Rate: 0.034050179211469536
Training Time: 0.004589557647705078
Prediction Time: 0.09273862838745117
```

BallTree

[89]:
```python
start=time.time()
knn_bt=KNeighborsClassifier(
    n_neighbors=optimal_k,
    weights=best_params['weights'],
    metric=best_params['metric'],
    algorithm='ball_tree'
)
knn_bt.fit(X_train_scaled,y_train)
train_time_bt=time.time()-start
start=time.time()
y_pred_bt=knn_bt.predict(X_test_scaled)
pred_time_bt=time.time()-start
```

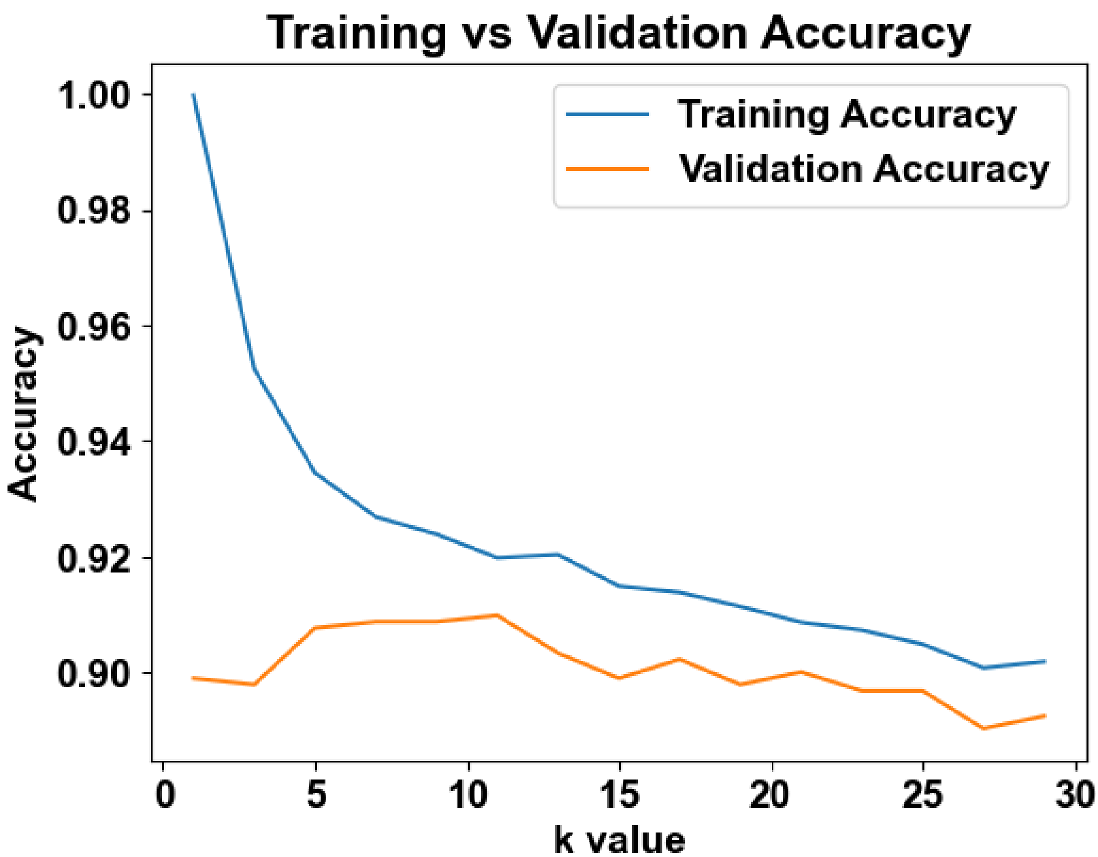
[90]:
```python
acc_bt,prec_bt,rec_bt,f1_bt,sp,fpr,cm=compute_metrics(y_test,y_pred_bt)
print("Final Balltree Metrics")
print("Accuracy:",acc_bt)
print("Precision:",prec_bt)
print("Recall:",rec_bt)
print("F1 Score:",f1_bt)
print("Specificity:",sp)
print("False Positive Rate:",fpr)
print("Training Time:",train_time)
print("Prediction Time:",pred_time)
```

```
Final Balltree Metrics
Accuracy: 0.9207383279044516
```

```
Precision: 0.9420731707317073
Recall: 0.8512396694214877
F1 Score: 0.894356005788712
Specificity: 0.9659498207885304
False Positive Rate: 0.034050179211469536
Training Time: 0.004589557647705078
Prediction Time: 0.09273862838745117
```

Training vs Validation

[92]:
```python
train_acc=[]
val_acc=[]
k_values=range(1,31,2)
for k in k_values:
    knn=KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train_scaled,y_train)
    train_acc.append(knn.score(X_train_scaled,y_train))
    val_acc.append(knn.score(X_test_scaled,y_test))
plt.plot(k_values,train_acc,label='Training Accuracy')
plt.plot(k_values,val_acc,label='Validation Accuracy')
plt.xlabel('k value')
plt.ylabel('Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.savefig("training_vs_validation_accuracy.png", dpi=300, bbox_inches="tight")
plt.show()
```



Cross Validation Scores

[102]:
```
gnb_scores=cross_val_score(
    GaussianNB(),
    X_train,
    y_train,
    cv=skf,
    scoring='precision'
)
```

[103]:
```
mnb_scores=cross_val_score(
    MultinomialNB(),
    X_train,
    y_train,
    cv=skf,
    scoring='precision'
)
```

[104]:
```
bnb_scores=cross_val_score(
    BernoulliNB(),
```

```
    X_train,
    y_train,
    cv=skf,
    scoring='precision'
)
```

[105]:

```
knn_kd_scores=cross_val_score(
    knn_kd,
    X_train_scaled,
    y_train,
    cv=skf,
    scoring='precision'
)
```

[106]:

```
knn_bt_scores=cross_val_score(
    knn_bt,
    X_train_scaled,
    y_train,
    cv=skf,
    scoring='precision'
)
```

One way ANOVA Test

[107]:

```
from scipy.stats import f_oneway
F_stat,p_value=f_oneway(
    gnb_scores,
    mnb_scores,
    bnb_scores,
    knn_kd_scores,
    knn_bt_scores
)

print("F-statistic:",F_stat)
print("p-value:",p_value)
```

```
F-statistic: 147.4451411146714
p-value: 1.537483731179684e-14
```

Mean Accuracy to find the best model

[108]:

```
print("Gaussian NB Mean precision:",gnb_scores.mean())
print("Multinomial NB Mean precision:",mnb_scores.mean())
print("Bernoulli NB Mean precision:",bnb_scores.mean())
print("KNN KDTree Mean precision:",knn_kd_scores.mean())
print("KNN BallTree Mean precision:",knn_bt_scores.mean())
```

```
Gaussian NB Mean precision: 0.7011869673414239
Multinomial NB Mean precision: 0.737146025168714
```

```
Bernoulli NB Mean precision: 0.8894670120013128
KNN KDTree Mean precision: 0.948881300360718
KNN BallTree Mean precision: 0.948881300360718
```

[109]:
```python
best_model=max(
    [
        ("Gaussian NB",gnb_scores.mean()),
        ("Multinomial NB",mnb_scores.mean()),
        ("Bernoulli NB",bnb_scores.mean()),
        ("KNN KDTree",knn_kd_scores.mean()),
        ("KNN BallTree",knn_bt_scores.mean())
    ],
    key=lambda x:x[1]
)

print("Best Model:",best_model)
```

```
Best Model: ('KNN KDTree', np.float64(0.948881300360718))
```

[ ]:

Naïve Bayes Performance Comparison

Metric	Gaussian NB	Multinomial NB	Bernoulli NB
Accuracy	0.8327	0.7763	0.8762
Precision	0.7145	0.7198	0.8716
Recall	0.9586	0.7079	0.8044
F1 Score	0.8188	0.7138	0.8366
Specificity	0.7508	0.7508	0.7508
Training Time (s)	0.0046	0.0092	0.0256

KNN Hyperparameter Tuning Results

Search Method	Best k	Best CV Accuracy	Best Parameters
Grid Search	9	0.9252	'metric': 'manhattan', 'weights': 'distance'
Randomized Search	6	0.9241	'metric': 'manhattan', 'weights': 'distance'

KNN Performance using Different Search Methods

Metric (KDTree)	Value
Optimal k	9
Accuracy	0.9207
Precision	0.9420
Recall	0.8512
F1 Score	0.8943
Training Time (s)	0.0056
Prediction Time (s)	0.0406

Metric (BallTree)	Value
Optimal k	9
Accuracy	0.9207
Precision	0.9420
Recall	0.8512
F1 Score	0.8943
Training Time (s)	0.0056
Prediction Time (s)	0.0406

KDTree vs BallTree Comparison

Criterion	KDTree	BallTree
Accuracy	0.9207	0.9207
Training Time (s)	0.0056	0.0056
Prediction Time (s)	0.0406	0.0406
Memory Usage	Low / Medium	Medium / High

Conclusion

Naïve Bayes provides fast and stable performance with high bias, whereas optimized KNN achieves better accuracy and generalization through careful hyperparameter tuning, with KDTree and Ball-Tree improving computational efficiency.

References

- Scikit-learn: Naïve Bayes
- Scikit-learn: KNN
- Scikit-learn: Hyperparameter Optimization
- Spambase Dataset