Data Mining HW3

Shardul Dabhane

Monday, 21st February 2022

1. (10 points) Given two strings s1 = abcdefabcdf, and s2 = bcdefaddcfa, compute their shingle vectors (using 3-shingles). Compute the Hamming distance and Jaccard similarity between the strings using their shingle vectors. Show the steps.

   I. Vectors of 3-shingles are given by:

   S1={(abc),(bcd),(cde),(def),(efa),(fab),(cdf)}
   S2={(bcd),(cde),(def),(efa),(fad),(add),(ddc),(def),(cfa)}

   II. Hamming Distance:

   Hamming Distance is the number of positions at which the corresponding symbols are different.
   To calculate Hamming Distance between the strings using their shingle vectors, we check the unique shingles for each string.
   For string s1, unique shingles are {(abc),(fab),(cdf)}.(count=3)
   For string s2, unique shingles are {(fad),(add),(ddc),(dcf),(cfa)}.(count=5)

   Therefore, the final Hamming distance would be 3+5=8.

   III. Jaccard Similarity:

   Jaccard Similarity=|S1∩S2| / |S1∪S2|

   S1∩S2={(bcd),(cde),(def),(efa)}
   |S1∩S2|=4

   S1∪S2={(abc),(bcd),(cde),(def),(efa),(fab),(cdf),(fad),(add),(ddc),(def),(cfa)}
   |S1∪S2|=12

   Hence, Jaccard Similarity=4/12=0.33
   JC=0.33

2. (25 points total) Given a shingle(word)-document matrix as below,

Shingle ID	d1	d2	d3	d4	d5	d6
0	0	1	0	1	1	0
1	0	0	1	0	0	0
2	1	1	0	0	0	0
3	1	1	0	1	1	1
4	1	0	0	1	1	1
5	0	1	0	0	1	1

(a) Jaccard similarity for all pairs:
For all Shingle IDs, we will compare values for the documents at those ids according to the formula:

Jaccard Similarity=$d1 \cap d2$ / $d1 U d2 = f_{11} / (f_{01}+f_{10}+f_{11})$

$f_{11:}$
The number of times 11 occurs between the 2 documents.

$f_{01:}$
The number of times 11 occurs between the 2 documents.

$f_{10:}$
The number of times 11 occurs between the 2 documents.

$f_{11:}$
The number of times 11 occurs between the 2 documents.

Let Jaccard Similarity=JS
Hence,
JS(d1,d2)=2/5=0.4
JS(d1,d3)=0/4=0
JS(d1,d4)=2/4=0.5
JS(d1,d5)=2/5=0.4
JS(d1,d6)=2/4=0.5

JS(d2,d3)=0/5=0
JS(d2,d4)=2/5=0.4

JS(d2,d5)=3/5=0.6
JS(d2,d6)=2/5=0.4

JS(d3,d4)=0/4=0
JS(d3,d5)=0/5=0
JS(d3,d6)=0/4=0

JS(d4,d5)=3/4=0.75
JS(d4,d6)=2/4=0.5

JS(d5,d6)=3/4=0.75

(b) The MinHash signatures for each column (document) :

We will compute the hash values by substituting the Shingle IDs for each hash function. From the obtained hash values, we will compute the MinHash signature for each permutation/ function.

Shingle ID	d1	d2	d3	d4	d5	d6	h1	h2	h3	h4
0	0	1	0	1	1	0	1	2	2	3
1	0	0	1	0	0	0	3	5	1	4
2	1	1	0	0	0	0	5	2	0	5
3	1	1	0	1	1	1	1	5	5	0
4	1	0	0	1	1	1	3	2	4	1
5	0	1	0	0	1	1	5	5	3	2

The MinHash signatures for a given function are the minimum values for the function where the document value is 1.

Signatures Matrix

	d1	d2	d3	d4	d5	d6
h1	1	1	3	1	1	1
h2	2	2	5	2	2	2
h3	0	0	1	2	2	3
h4	0	0	4	0	0	0

(c) Compute the similarities for all pairs of the six documents using the signatures

Let SS be used to denote the signature similarity.

SS(d1,d2)=4/4=1
SS(d1,d3)=0/4=0

SS(d1,d4)=3/4=0.75
SS(d1,d5)=3/4=0.75
SS(d1,d6)=3/4=0.75

SS(d2,d3)=0/4=0
SS(d2,d4)=3/4=0.75
SS(d2,d5)=3/4=0.75
SS(d2,d6)=3/4=0.75

SS(d3,d4)=0/4=0
SS(d3,d5)=0/4=0
SS(d3,d6)=0/4=0

SS(d4,d5)=4/4=1
SS(d4,d6)=3/4=0.75

SS(d5,d6)=3/4=0.75

(d) Does MinHash provide good signatures for computing the document similarity for this example (2 points)?

Comparing the Jaccard Similarity of all documents with Signature Similarity of all documents, we find that except for a few documents, most of the values match each other or are close to each other. **Hence, we can say that Minhash provides good signatures for computing the document similarity in this example**.

(e) Can you design another hash function that provides true permutation (3 points)?

Hash function with true permutation can be derived as,

$h_{a,b}(x) = ((ax + b) \bmod p) \bmod N$
Where, a, b are random integers and p is a prime number such that (p > N)

Thus, hash function for given example is,
h1(x) = ((7x + 3) % 11) % 6

3. (15 points) Credit card fraud detection using KNN. Please use this code as the start code.

I. **Data preprocessing is performed in the given code. We check if there are any missing values in the dataset. Then, the data is normalized using Standard Scalar.**

II. **The distance metric used in the original code is the Minkowski distance metric with the value of p=2.**

III. In the given example, the area under the ROC curve is used as a performance metric. The AUC - ROC curve is a performance measure for classification problems with various threshold values. By definition, ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes.

The higher the AUC, the better the model is at predicting 0 classes like 0 and 1 classes as 1. Consequently, the Higher the AUC, the better the model is at distinguishing between classes.

Following is the summary of different values of AUC ROC for some distance metrics and different values of k. Please find the corresponding Kaggle code changes in HW3.ipynb

Distance Metric	k=5	k=10	k=50	k=100
Minkowski	0.578	0.580	0.645	0.653
Euclidean	0.578	0.580	0.645	0.653
Manhattan	0.638	0.647	0.692.	0.729

From the above statistics of the AUC-ROC curve for different distance metrics and different k values, we can conclude that,

i. **Model performance increases with the increase in the value of k.**

ii. **Among the distance metrics tested, the Manhattan distance metric performed the bes**t.

iii**. Furthermore, by definition, Minkowski with p=2 is equivalent to Standard Euclidean metric, the same is evident with the results**.

4. (25 points total) Link analysis

(a) Given the above toy web (with 6 web pages, A, B, · · ·, F), derive its transition probability matrix (5 points).

The Transition Probability Matrix is:

	a	**b**	**c**	**d**	**e**	**f**
a	**0**	**1/4**	**0**	**0**	**0**	**1/2**
b	**1/2**	**0**	**0**	**0**	**0**	**0**
c	**0**	**1/4**	**0**	**0**	**1/2**	**1/2**
d	**0**	**1/4**	**1**	**0**	**0**	**0**
e	**1/2**	**0**	**0**	**1/2**	**0**	**0**
f	**0**	**1/4**	**0**	**1/2**	**1/2**	**0**

(b) Assume a surfer is on web page A, what's the probability that the person will be next visiting B and then D (5 points)?

The Probability of a person going from A to B, P(A $\Longrightarrow$ B)= 1/2.

Probability of person going from B to D, P(B $\Longrightarrow$ D)= 1/4.

Thus, the Probability of a person going from A to B to D=
P(A $\Longrightarrow$ B) × P(B $\Longrightarrow$ D) = 1/2*1/4 = 1/8 = 0.125

(c) Implementation of Page Rank Algorithm and KNN on Notebook.

5. Write a summary for this review article: Information Security in Big Data: Privacy and Data Mining. [25 pts](on final page).

Shardul Dabhane

Problem 3: Changes in Kaggle Code for KNN

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

#in Kaggle, File -> Add or upload data -> search for credit card
#note about the folder: ../input/creditcard
#change the folder if you have data in a different folder
data = pd.read_csv("creditcard.csv")
print(data.shape)
#data.head()
data.describe()
```

```
(284807, 31)
```

Out[1]:

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8480
mean	94813.859575	3.919560e-15	5.688174e-16	-8.769071e-15	2.782312e-15	-1.552563e-15	2.010663e-15	-1.694249e-15	-1.927(
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.1943
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.3216
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.0862
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.2358
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.2734
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.0007

8 rows x 31 columns

In [2]:

```
#check if there are missing data
data.isnull().any().any()

#change 'Class' dtype to "bool"
data['Class'] = data['Class'].astype('bool')
```

In [3]:

```
class_zero = data.Class.value_counts().values[0]
class_one = data.Class.value_counts().values[1]
print(data["Class"].value_counts())
```

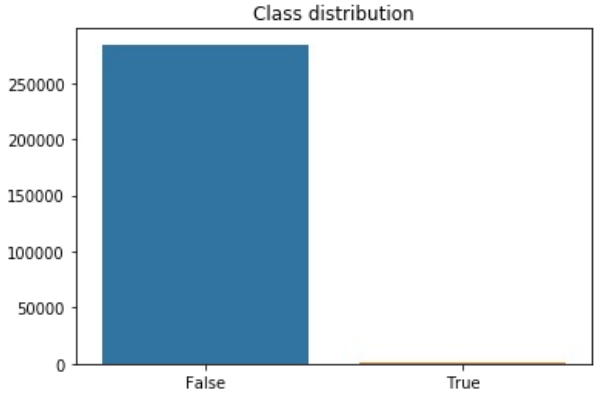
```
False    284315
True        492
Name: Class, dtype: int64
```

In [4]:

```
sb.barplot(x=data.Class.value_counts().index.values, y=data.Class.value_counts().values)
plt.title("Class distribution")
```

Out[4]:

```
Text(0.5, 1.0, 'Class distribution')
```



In [5]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
data['AmountNormalized'] = StandardScaler().fit_transform(data['Amount'].values.reshape(-1, 1))
data['AmountNormalized'].describe()
```

Out[5]:

```
count    2.848070e+05
mean     3.202236e-16
std      1.000002e+00
min     -3.532294e-01
25%     -3.308401e-01
50%     -2.652715e-01
75%     -4.471707e-02
max      1.023622e+02
Name: AmountNormalized, dtype: float64
```

In [6]:

```
X = data.iloc[:, data.columns != 'Class'].values
y = data.iloc[:, data.columns == 'Class'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

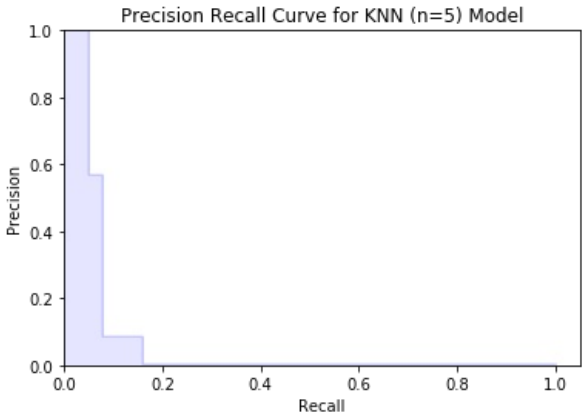
In [7]:

```
def plot_precision_recall_curve(y_actual, y_score, model_name):
    precision, recall, _ = metrics.precision_recall_curve(y_actual, y_score)
    curve_data = pd.DataFrame(columns = range(0, len(precision)))
    curve_data.loc['Precision'] = precision
    curve_data.loc['Recall'] = recall
    #print (curve_data)
    plt.step(recall, precision, color='b', alpha=0.1, where='post')
    plt.fill_between(recall, precision, step='post', alpha=0.1, color='b')
    plt.title('Precision Recall Curve for {} Model'.format(model_name))
    plt.xlabel('Recall')
    plt.ylabel('Precision')
    plt.xlim([0, 1.05])
    plt.ylim([0, 1.0])

def evaluate_model(y_actual, y_pred, y_score, model_name):
    cm = metrics.confusion_matrix(y_actual, y_pred)
    print ('Confusion Matrix for {} Model'.format(model_name))
    print (cm)
    print ('Classification Report for {} Model'.format(model_name))
    print (metrics.classification_report(y_actual, y_pred, digits=6))
    print ('Area under under ROC curve for {} Model'.format(model_name))
    print (metrics.roc_auc_score(y_actual, y_score))
    plot_precision_recall_curve(y_actual, y_score, model_name)
```

In [8]:

```
#KNN with minkowski metric, k=5 and p=2
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=5, metric= 'minkowski', p=2)
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=5)')
```

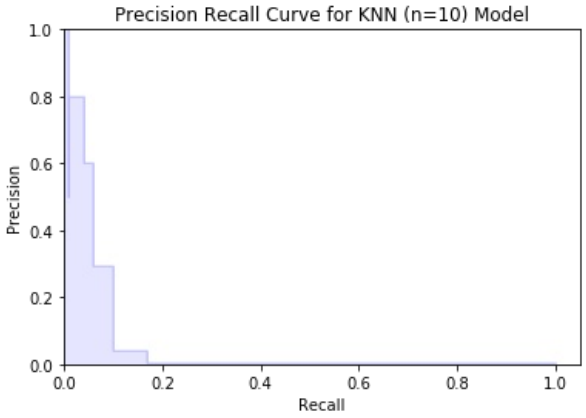
```
Confusion Matrix for KNN (n=5) Model
[[56861     0]
 [   96     5]]
Classification Report for KNN (n=5) Model
              precision    recall  f1-score   support

       False   0.998315  1.000000  0.999157     56861
        True   1.000000  0.049505  0.094340       101

    accuracy                       0.998315     56962
   macro avg   0.999157  0.524752  0.546748     56962
weighted avg   0.998318  0.998315  0.997552     56962

Area under under ROC curve for KNN (n=5) Model
0.5777933195088735
```



In [9]:

```
#KNN with minkowski metric, k=10 and p=2

from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=10, metric= 'minkowski', p=2)
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=10)')
```

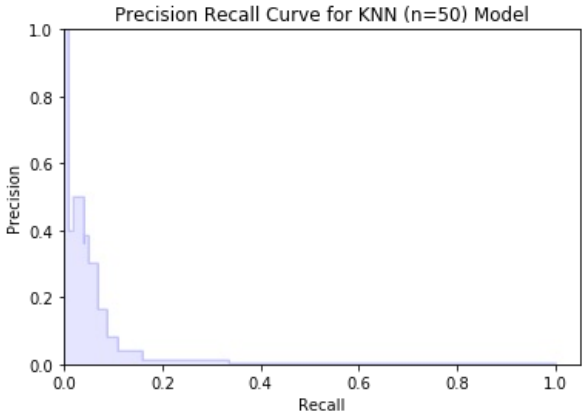
```
Confusion Matrix for KNN (n=10) Model
[[56861     0]
 [  100     1]]
Classification Report for KNN (n=10) Model
              precision    recall  f1-score   support

       False   0.998244  1.000000  0.999121     56861
        True   1.000000  0.009901  0.019608       101

    accuracy                       0.998244     56962
   macro avg   0.999122  0.504950  0.509365     56962
weighted avg   0.998248  0.998244  0.997385     56962

Area under under ROC curve for KNN (n=10) Model
0.580813016142718
```



In [10]:

```
#KNN with minkowski metric, k=50 and p=2

from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=50, metric= 'minkowski', p=2)
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=50)')
```

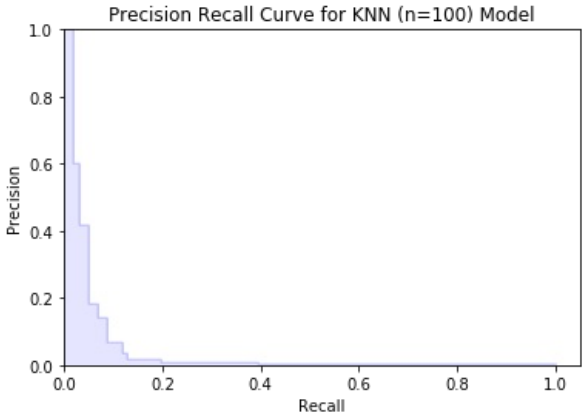
```
Confusion Matrix for KNN (n=50) Model
[[56861     0]
 [  101     0]]
Classification Report for KNN (n=50) Model
              precision    recall  f1-score   support

       False   0.998227  1.000000  0.999113     56861
        True   0.000000  0.000000  0.000000       101

    accuracy                       0.998227     56962
   macro avg   0.499113  0.500000  0.499556     56962
weighted avg   0.996457  0.998227  0.997341     56962

Area under under ROC curve for KNN (n=50) Model
0.6450030741981359

C:\Users\Shardul\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [11]:

```
#KNN with minkowski metric, k=100 and p=2
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=100, metric= 'minkowski', p=2)
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=100)')
```

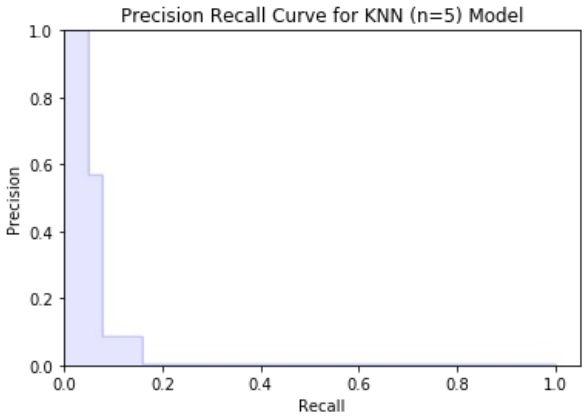
```
Confusion Matrix for KNN (n=100) Model
[[56861     0]
 [  101     0]]
Classification Report for KNN (n=100) Model
              precision    recall  f1-score   support

       False   0.998227  1.000000  0.999113     56861
        True   0.000000  0.000000  0.000000       101

    accuracy                       0.998227     56962
   macro avg   0.499113  0.500000  0.499556     56962
weighted avg   0.996457  0.998227  0.997341     56962

Area under under ROC curve for KNN (n=100) Model
0.6531097808256054

C:\Users\Shardul\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [12]:

```
# KNN with k=5 and metric='euclidean'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=5, metric= 'euclidean')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=5)')
```

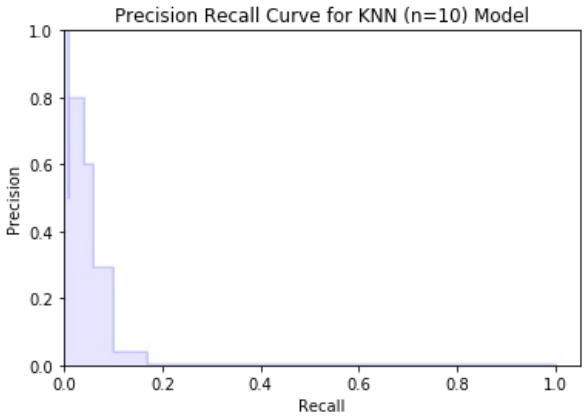
```
Confusion Matrix for KNN (n=5) Model
[[56861     0]
 [   96     5]]
Classification Report for KNN (n=5) Model
              precision    recall  f1-score   support

       False   0.998315  1.000000  0.999157     56861
        True   1.000000  0.049505  0.094340       101

    accuracy                       0.998315     56962
   macro avg   0.999157  0.524752  0.546748     56962
weighted avg   0.998318  0.998315  0.997552     56962

Area under under ROC curve for KNN (n=5) Model
0.5777933195088735
```



In [13]:

```
# KNN with k=10 and metric='euclidean'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=10, metric= 'euclidean')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=10)')
```

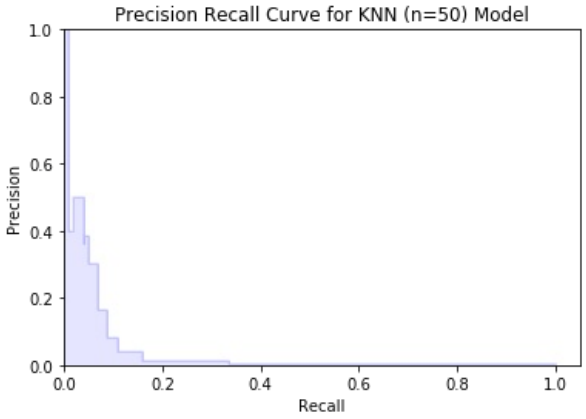
```
Confusion Matrix for KNN (n=10) Model
[[56861     0]
 [  100     1]]
Classification Report for KNN (n=10) Model
              precision    recall  f1-score   support

       False   0.998244  1.000000  0.999121     56861
        True   1.000000  0.009901  0.019608       101

    accuracy                       0.998244     56962
   macro avg   0.999122  0.504950  0.509365     56962
weighted avg   0.998248  0.998244  0.997385     56962

Area under under ROC curve for KNN (n=10) Model
0.580813016142718
```



In [14]:

```
# KNN with k=50 and metric='euclidean'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=50, metric= 'euclidean')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=50)')
```

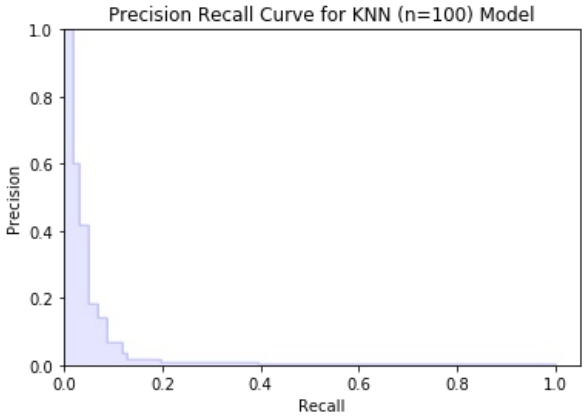
```
Confusion Matrix for KNN (n=50) Model
[[56861     0]
 [  101     0]]
Classification Report for KNN (n=50) Model
              precision    recall  f1-score   support

       False   0.998227  1.000000  0.999113     56861
        True   0.000000  0.000000  0.000000       101

    accuracy                       0.998227     56962
   macro avg   0.499113  0.500000  0.499556     56962
weighted avg   0.996457  0.998227  0.997341     56962

Area under under ROC curve for KNN (n=50) Model
0.6450030741981359

C:\Users\Shardul\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [15]:

```
# KNN with k=100 and metric='euclidean'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=100, metric= 'euclidean')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=100)')
```

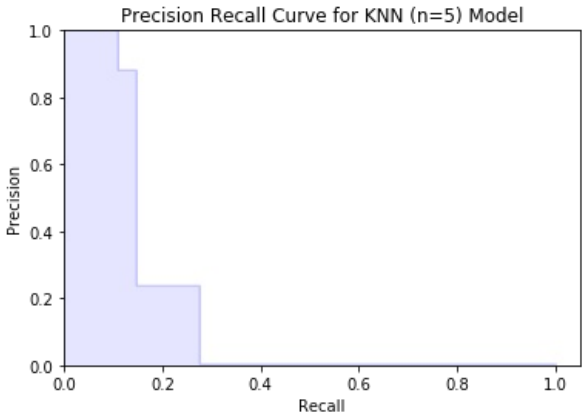
```
Confusion Matrix for KNN (n=100) Model
[[56861     0]
 [  101     0]]
Classification Report for KNN (n=100) Model
              precision    recall  f1-score   support

       False   0.998227  1.000000  0.999113     56861
        True   0.000000  0.000000  0.000000       101

    accuracy                       0.998227     56962
   macro avg   0.499113  0.500000  0.499556     56962
weighted avg   0.996457  0.998227  0.997341     56962

Area under under ROC curve for KNN (n=100) Model
0.6531097808256054

C:\Users\Shardul\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [16]:

```
# KNN with k=5 and metric='manhattan'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=5, metric= 'manhattan')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=5)')
```

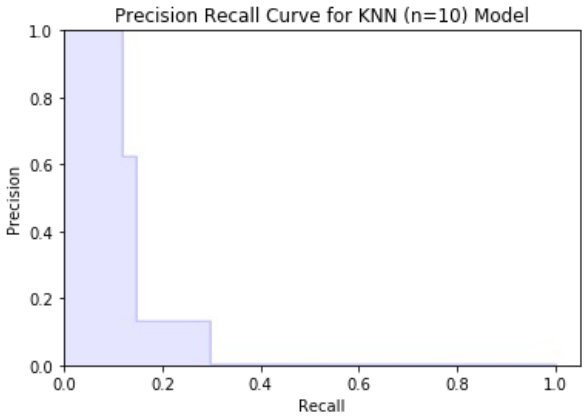
```
Confusion Matrix for KNN (n=5) Model
[[56861     0]
 [   90    11]]
Classification Report for KNN (n=5) Model
              precision    recall  f1-score   support

       False   0.998420  1.000000  0.999209     56861
        True   1.000000  0.108911  0.196429       101

    accuracy                       0.998420     56962
   macro avg   0.999210  0.554455  0.597819     56962
weighted avg   0.998422  0.998420  0.997786     56962

Area under under ROC curve for KNN (n=5) Model
0.6379445202570591
```



In [17]:

```
# KNN with k=10 and metric='manhattan'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=10, metric= 'manhattan')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=10)')
```

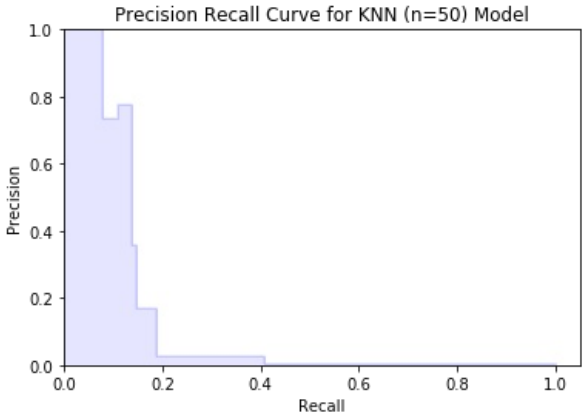
```
Confusion Matrix for KNN (n=10) Model
[[56861     0]
 [   93     8]]
Classification Report for KNN (n=10) Model
              precision    recall  f1-score   support

       False   0.998367  1.000000  0.999183     56861
        True   1.000000  0.079208  0.146789       101

    accuracy                       0.998367     56962
   macro avg   0.999184  0.539604  0.572986     56962
weighted avg   0.998370  0.998367  0.997671     56962

Area under under ROC curve for KNN (n=10) Model
0.6470332116133124
```



In [18]:

```
# KNN with k=50 and metric='manhattan'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=50, metric= 'manhattan')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=50)')
```

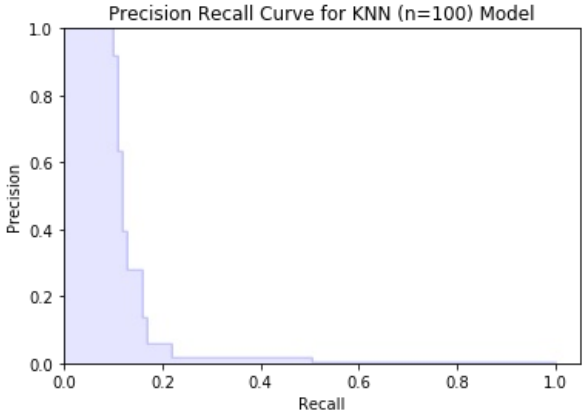
```
Confusion Matrix for KNN (n=50) Model
[[56861     0]
 [  101     0]]
Classification Report for KNN (n=50) Model
              precision    recall  f1-score   support

       False   0.998227  1.000000  0.999113     56861
        True   0.000000  0.000000  0.000000       101

    accuracy                       0.998227     56962
   macro avg   0.499113  0.500000  0.499556     56962
weighted avg   0.996457  0.998227  0.997341     56962

Area under under ROC curve for KNN (n=50) Model
0.6924904243647136

C:\Users\Shardul\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```



In [19]:

```
# KNN with k=100 and metric='manhattan'
from sklearn.neighbors import KNeighborsClassifier
#train
knn = KNeighborsClassifier(n_neighbors=100, metric= 'manhattan')
knn.fit(X_train, y_train.ravel())
#test
y_pred_knn = knn.predict(X_test)
y_prob_knn = knn.predict_proba(X_test)

evaluate_model(y_test, y_pred_knn, y_prob_knn[:, [1]], 'KNN (n=100)')
```

```
Confusion Matrix for KNN (n=100) Model
[[56861     0]
 [  101     0]]
Classification Report for KNN (n=100) Model

C:\Users\Shardul\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedMetri
cWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted sam
ples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

              precision    recall  f1-score   support

       False   0.998227  1.000000  0.999113     56861
        True   0.000000  0.000000  0.000000       101

    accuracy                       0.998227     56962
   macro avg   0.499113  0.500000  0.499556     56962
weighted avg   0.996457  0.998227  0.997341     56962

Area under under ROC curve for KNN (n=100) Model
0.7287805889679557
```



Question 4: Link Analysis and PageRank algorithm implementation

In [20]:

```
#Pagerank implementation.

#use numpy module
import numpy as np

# Define the function for page rank algorithm
def page_rank(transition_matrix):

    # Define the initial rank matrix
    rank_matrix = np.ones(6)/6

    # Get the dot product of the given matrix with rank
    dot_product = transition_matrix.dot(rank_matrix)

    # Repeat loop until ranks converge
    while True:
        dot_product = transition_matrix.dot(rank_matrix)
        if np.linalg.norm(np.abs(rank_matrix - dot_product)) < 0.01:
            break
        rank_matrix = dot_product

    return dot_product
```

In [21]:

```
# Given transition matrix as per the problem
transition_matrix = np.array([[ 0,  0.25,  0,  0,  0,  0.5],
       [ 0.5,  0,  0,  0,  0,  0],
       [ 0,  0.25,  0,  0,  0.5,  0.5],
       [ 0,  0.25,  1.0,  0,  0,  0],
       [ 0.5,  0,  0,  0.5,  0,  0],
       [ 0,  0.25,  0,  0.5,  0.5,  0]])
ranks_of_pages = page_rank(transition_matrix)
ranks_of_pages
```

Out[21]:

```
array([0.12304687, 0.06119792, 0.20898438, 0.22591146, 0.17057292,
       0.21028646])
```

Page ranks of web pages here after implementing the pagerank algorithm:

1. D(0.22591146)

2. F(0.21028646)

3. C(0.20898438)

4. E(0.17057292)

5. A(0.12304687)

6. B(0.06119792)

In [22]:

```
# Defining the function for page rank algorithm with different distribution

def page_rank_scaled(transition_matrix):
    # Scaled rank matrix initial distribution
    rank_matrix = 100 * np.ones(6)/6

    # Get the dot product of the given matrix with rank
    dot_product = transition_matrix.dot(rank_matrix)

    # Repeat loop until ranks converge
    while True:
        dot_product = transition_matrix.dot(rank_matrix)

        if np.linalg.norm(np.abs(rank_matrix - dot_product)) < 0.001:
            break
        rank_matrix = dot_product
    return dot_product
```

In [23]:

```
ranks_of_pages = page_rank_scaled(transition_matrix)
ranks_of_pages
```

Out[23]:

```
array([12.18261859,  6.0914689 , 20.81192096, 22.33527407, 17.25902329,
       21.31969419])
```

In [25]:

```
# Defining the function for page rank algorithm with different distribution

def page_rank_scaled_1(transition_matrix):
    # Scaled rank matrix initial distribution
    rank_matrix = 50 * np.ones(6)/6

    # Get the dot product of the given matrix with rank
    dot_product = transition_matrix.dot(rank_matrix)

    # Repeat loop until ranks converge
    while True:
        dot_product = transition_matrix.dot(rank_matrix)

        if np.linalg.norm(np.abs(rank_matrix - dot_product)) < 0.001:
            break
        rank_matrix = dot_product
    return dot_product

ranks_of_pages = page_rank_scaled_1(transition_matrix)
ranks_of_pages
```

Out[25]:

```
array([ 6.09123373,  3.04578759, 10.40580814, 11.16776397,  8.62960096,
       10.65980561])
```

Conclusion : The rank values for each page get scaled with scaling of rank matrix and changing the initial distribution but the overall ranks remain same with D having highest rank and F, C, E, A, B in the decreasing order.

References:

[1] Introduction to Data Mining 2nd Edition By Tan, Steinbach, Kumar,Karpatne

[2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html)

[3] https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5)

[4] https://notebook.community/hktxt/MachineLearning/PageRank (https://notebook.community/hktxt/MachineLearning/PageRank)

[5] https://developpaper.com/numerical-analysis-power-iteration-and-pagerank-algorithm/ (https://developpaper.com/numerical-analysis-power-iteration-and-pagerank-algorithm/)

In [ ]:

Information Security in Big Data: Privacy and Data Mining LEI XU, CHUNXIAO JIANG, (Member, IEEE), JIAN WANG, (Member, IEEE), JIAN YUAN, (Member, IEEE), AND YONG REN, (Member, IEEE)

The popularity of data mining technologies is a serious threat to the privacy of individual users. To combat this, research is being done in privacy-preserving data mining (PPDM). The basic idea behind PPDM is to modify the data using data mining algorithms without undermining data security. Current studies in PPDM focus on reducing security breaches by data mining operations, while there is still a substantial risk to sensitive information during data collection, data publishing, and information delivery. The paper looks at ways to protect data while performing data mining algorithms from a broad perspective. The paper identifies four different types of users in data mining applications: data provider, data collector, data miner, and decision-maker, and then looks at each user's unique concern regarding data privacy and how it should be protected during data mining. The paper also deals with interactions among different users during data-mining operations.

A data provider is someone who provides data to a data collector or someone who provides data to a data miner. A data provider's concerns regarding privacy would be what kind of and how much information other people can obtain from his data. He can refuse to provide data that may contain sensitive information about him. He can also trade his private data for better service or monetary benefits. If both the previous methods are possible, the data provider can distort his data so that the information cannot be easily disclosed. It is complicated to protect a data provider's data, as it might disrupt the entire functionality of data mining. He should be smart enough to negotiate with the data collector so that every sensitive information he provides leads to benefits. A data provider should also be helped in being able to easily detect when his sensitive information has been leaked.

A data collector is someone who collects data from data providers for further data mining operations. If data providers don't take sufficient care while giving data to data collectors, it might be disclosed to the wrong entities. From auxiliary information, some people can easily gain access to an individual's private records. A data collector should modify the data he receives from data providers before releasing it to others. The modifications made by him should not affect data utility. The data modification process adopted by the data collector, to preserve privacy and utility simultaneously is called privacy-preserving data publishing (PPDP). One of the main methods in this approach is to have the data anonymized to protect an individual's identity. This can be achieved with generalization, suppression, anatomization, and other such methods, which ensure the removal of identifiers of individuals. For social network data, which is graph-based, we can use edge modification and other methods. As ''personalization'' becomes popular in current data-driven applications, issues related to personalized data anonymization, like how to formulate personalized privacy preferences in a more flexible way and how to obtain such preference with less effort paid by data providers, need further investigation.

A data miner is someone who performs data mining tasks on the data. He has to ensure that during data mining, the data provider's sensitive information is not leaked. PPDM approaches to deal with this problem based on data mining tasks are privacy-preserving association rule mining, privacy-preserving classification, and privacy-preserving clustering.

Privacy trouble for a data miner can also happen due to the release of the learning model used on the data and collaboration with other data miners. The best approach for a data-miner to deal with security issues of data would be to modify the data.

A decision-maker is one who makes the final decisions on the data from data mining to solve certain problems. While most of the privacy concerns are dealt with by the previous 3 participants in the data mining, process, the decision-maker also has to ensure that the data mining results are not leaked to competitors. He also has to evaluate the credibility of the data mining results. To determine the trustworthiness of data, data provenance is used. It refers to the information that helps determine the derivation history of the data, starting from the original source. To determine the credibility of the information we get, we need to look into the preparation of the dataset and feature selection. It is still difficult to identify fake information and we need to take methods from multiple disciplines like natural language processing, data mining, machine learning, social networking analysis, and information provenance, into the mix.

The paper also looks into game theory, a formal approach to model situations in which the participants in the data mining process have to choose the most optimal action to take with taking the effects the action will have on the other participants into consideration. All participants should look to ensure that they are benefited themselves while helping to solve the larger data mining problem.

If not by technical methods, we can achieve data privacy in data mining by legislation by various countries and following industry conventions. The paper also proposes future research directions that can be followed by introducing more PPDM algorithms, looking into data customization or reverse data management(RDM), and developing new provenance models that specify what kind of provenance information is required and how to use that information.