K - Nearest Neighbors (K-NN)

Aim:

1. Implement the K-NN algorithm for classification or regression.
2. Apply the K-NN algorithm to a given dataset and predict the class or value for test data.
3. Evaluate the accuracy or error of the predictions and analyze the results

Theory:

K-Nearest Neighbors (K-NN) is a simple but powerful machine learning algorithm used for both classification and regression tasks.

1. Intuition: K-NN is based on the idea that objects (data points) that are close to each other in a feature space are more likely to belong to the same class or have similar values (for regression).

2. How it works:
 - Classification: Given a new data point, K-NN finds the K nearest data points in the training dataset and assigns the class label that is most common among those K neighbors to the new point.
 - Regression: For regression tasks, K-NN calculates the average (or another aggregation) of the target values of the K nearest neighbors and assigns this value to the new data point.

3. Hyperparameter K: The choice of the hyperparameter K (the number of neighbors to consider) is critical. A small K may lead to a noisy model (sensitive to outliers), while a large K may lead to a biased model (smoothing over variations in the data). K is typically an odd number to avoid ties in voting.

4. Distance Metric: K-NN uses a distance metric (e.g., Euclidean distance, Manhattan distance, etc.) to measure the similarity between data points. The choice of distance metric should be appropriate for your data and problem.

5. Scaling Features: It's important to scale features before applying K-NN, especially when using distance-based metrics, to ensure that all features have equal influence on the results.

6. Pros:
 - Simple and easy to understand.
 - No assumptions about the data distribution.
 - Works well for both classification and regression tasks.
 - Non-parametric (does not make assumptions about the functional form of relationships).

7. Cons:
 - Can be computationally expensive, especially for large datasets.
 - Sensitive to the choice of K and the distance metric.
 - Requires a sufficient amount of training data.
 - May not perform well when the feature space is high-dimensional.

8. Use Cases:

	- K-NN is often used for tasks such as recommendation systems, image classification, and anomaly detection. - It can be used as a baseline model for comparison with more complex algorithms. 9. Model Evaluation: Common evaluation metrics for K-NN include accuracy (for classification) and mean squared error (for regression). Cross-validation is often used to estimate the model's generalization performance. 10. Implementation: Python libraries like scikit-learn provide easy-to-use implementations of K-NN for both classification and regression. In summary, K-NN is a versatile and interpretable algorithm suitable for various tasks. It's important to choose appropriate values for K and the distance metric based on your data and problem domain to achieve good performance.

Dataset:	We use the iris dataset which is available in the public domain, although we can also create our own datasets We download the iris dataset and save in .csv format

The following is the Python code to implement K-NN Classifier

(To run this code, Scikit-Learn must be installed (pip install scikit-learn) and Pandas must be installed (pip install pandas).

Python Code	

Output	

For Video demonstration of the practical click on the link or scan the QR-code

https://youtu.be/_tcjQjh2rEE	

PRACTICAL NO: 8

Aim:- Implement the K-NN Algorithm for classification or regression.
Apply K-NN Algorithm on the given dataset & predict the class or value for test data.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('ggplot')
```

In [2]:
```
df = pd.read_csv('C:/Users/RDNC/Desktop/diabetes.csv')
df.head()
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

In [3]:
```
df.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```
df.dtypes
```

Out[4]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [9]:
```
x= df.drop('Outcome',axis=1).values
y = df['Outcome'].values
```

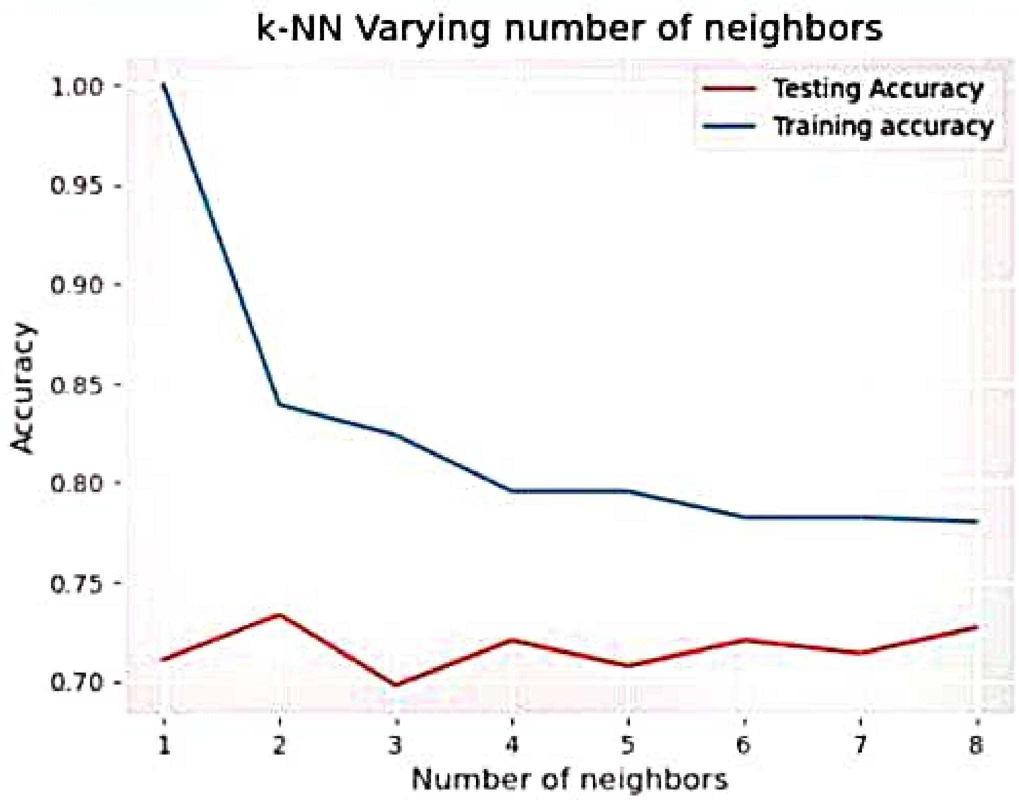
In [10]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.4,random_state=42, st
```

In [24]:
```
from sklearn.neighbors import KNeighborsClassifier
neighbors = np.arange(1,9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

for i,k in enumerate(neighbors):
#Setup a knn classifier with k neighbors
    knn = KNeighborsClassifier(n_neighbors=k)
#Fit the model
    knn.fit(X_train, y_train)
#Compute accuracy on the training set
```

```
    train_accuracy[i] = knn.score(X_train, y_train)
#Compute accuracy on the test set
    test_accuracy[i] = knn.score(X_test, y_test)
```

In [25]:
```
plt.title('k-NN Varying number of neighbors')
plt.plot(neighbors, test_accuracy, label='Testing Accuracy')
plt.plot(neighbors, train_accuracy, label='Training accuracy')
plt.legend()
plt.xlabel('Number of neighbors')
plt.ylabel('Accuracy')
plt.show()
```

In [27]:
```
knn = KNeighborsClassifier(n_neighbors=7)
```

In [30]:
```
knn.fit(X_train,y_train)
```

Out[30]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [32]:
```
knn.score(X_test,y_test)
```

```
C:\Users\RDNC\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: F
utureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default
behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this
behavior will change: the default value of `keepdims` will become False, the `axis` o
ver which the statistic is taken will be eliminated, and the value None will no longe
r be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

Out[32]:
```
0.7142857142857143
```

In [36]:
```
from sklearn.metrics import confusion_matrix
```

In [ ]:

In [37]:
```
y_pred = knn.predict(X_test)
```

In [39]:
```
confusion_matrix(y_test,y_pred)
```

Out[39]:
```
array([[163,  43],
       [ 45,  57]], dtype=int64)
```

In [43]:
```
from sklearn.metrics import classification_report
```

In [ ]:

In [44]:
```
print(classification_report(y_test,y_pred))
```

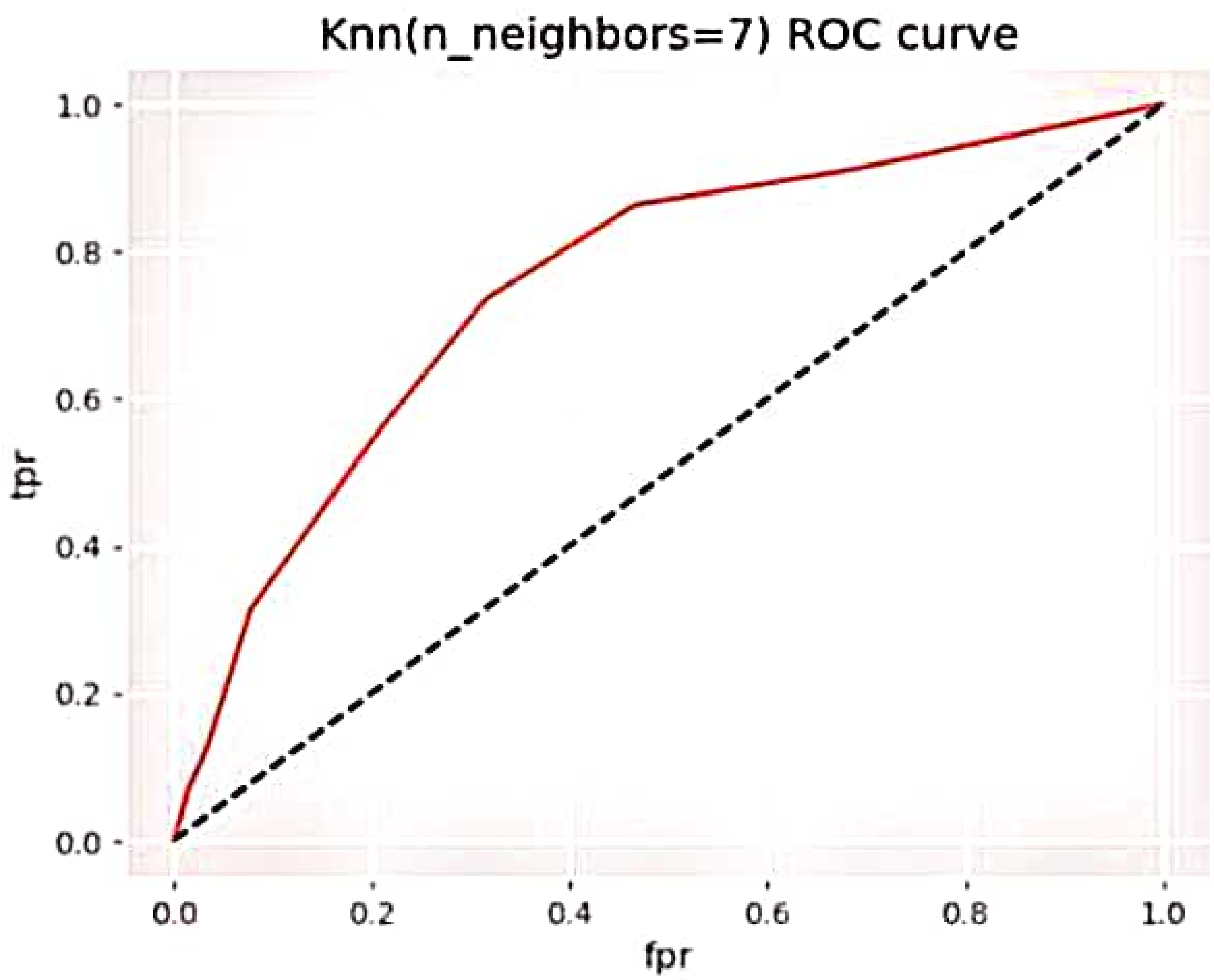
```
              precision    recall  f1-score   support

           0       0.78      0.79      0.79       206
           1       0.57      0.56      0.56       102

    accuracy                           0.71       308
   macro avg       0.68      0.68      0.68       308
weighted avg       0.71      0.71      0.71       308
```

In [46]:
```
y_pred_proba = knn.predict_proba(X_test)[:,1]
```

In [48]:
```
from sklearn.metrics import roc_curve
```

In [50]:
```
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
```

In [52]:
```
plt.plot([0,1],[0,1],'k--')
plt.plot(fpr,tpr, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Knn(n_neighbors=7) ROC curve')
plt.show()
```




```
In [54]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,y_pred_proba)

Out[54]: 0.7536645726251665

In [56]: from sklearn.model_selection import GridSearchCV

In [58]: param_grid = {'n_neighbors':np.arange(1,50)}

In [61]: knn = KNeighborsClassifier()
         knn_cv= GridSearchCV(knn,param_grid,cv=5)
         knn_cv.fit(x,y)

]: GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
              param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 1
0, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])})

]: knn_cv.best_score_

]: 0.7578558696205755

]: knn_cv.best_params_

]: {'n_neighbors': 14}
```