CS613 Project -Francis Kurian

K-Nearest Neighbors (KNN) project

Objective:

1. investigate how different values of k and distance metrics affect the accuracy of a KNN classifier on a specific dataset.
2. Visualize the decision boundaries for k iterations

Steps:

1.Select a Dataset from UCI depository, preprocess data to handle missing values etc.

2.Implement a KNN Algorithm: for example Minkowski distance metric. (Use libraries like scikit-learn)

3.Experiment with Different k Values: Conduct experiments with various values of k and observe how the choice of k affects the performance of the model. Plot accuracy for each k value to visualize the results.

4.Experiment with Different Distance Metrics: Metrics such as Euclidean distance, Manhattan distance, and Minkowski distance with various p values (power parameter) to observe how the choice of distance metric influences the model's performance.

5.Creat a synthetic two variable dataset to demonstrate decision boundaries of KNN at various iterations of k.

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving test.csv to test.csv
```

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
pip install ucimlrepo
```

```
from ucimlrepo import fetch_ucirepo
# fetch dataset
heart_disease = fetch_ucirepo(id=45)
# data (as pandas dataframes)
X = heart_disease.data.features
y = heart_disease.data.targets
```

```
combined_df = pd.concat([X, y], axis=1)
combined_df = combined_df.dropna()
combined_df.reset_index(drop=True, inplace=True)
```

```
combined_df.head(5)
```

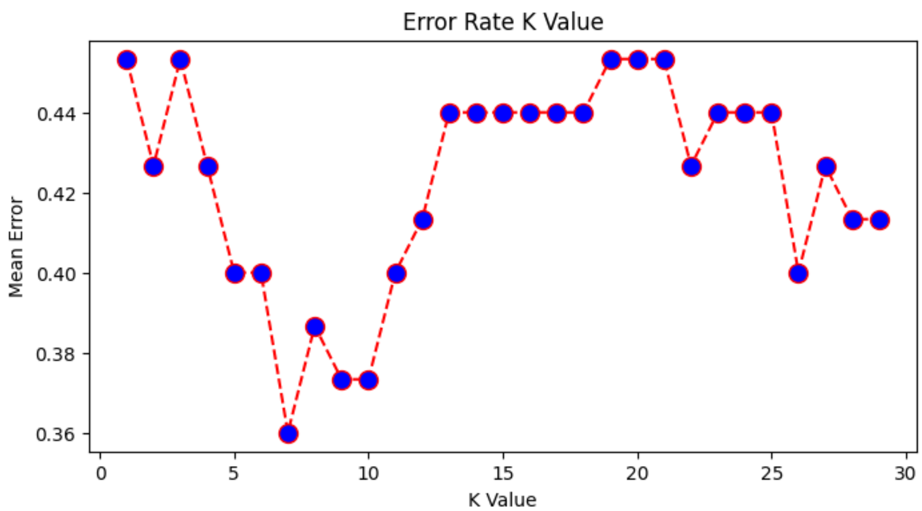
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	num
0	63	1	1	145	233	1	2	150	0	2.3	3	0.0	6.0	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3.0	3.0	2
2	67	1	4	120	229	0	2	129	1	2.6	2	2.0	7.0	1
3	37	1	3	130	250	0	0	187	0	3.5	3	0.0	3.0	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0.0	3.0	0

```
from sklearn.model_selection import train_test_split
x= combined_df.iloc[:,0:13].values
y= combined_df['num'].values

x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
```

```
x_train= st_x.fit_transform(x_train)
```

```
error = []
# Calculating error for K values between 1 and 30
for i in range(1, 30):
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))
plt.figure(figsize=(8, 4))
plt.plot(range(1, 30), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
print("Minimum error:-",min(error),"at K =",error.index(min(error))+1)
plt.show()
```

```
Minimum error:- 0.36 at K = 7
```

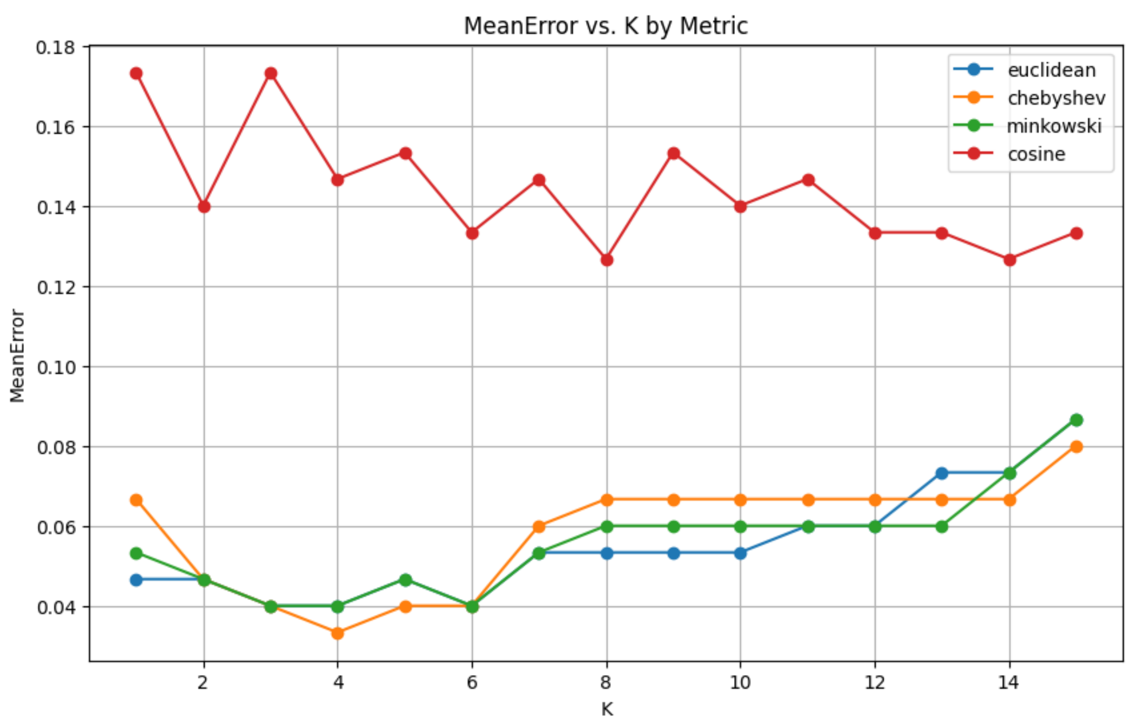


Experiment with various k values and Distance Metrics on UCI Health Dataset

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Create empty lists to store error, accuracy, and metrics
error = []
accuracy = []
metrics_list = []

# Create a DataFrame to store results
results_df = pd.DataFrame(columns=['K', 'Mean Error', 'Accuracy', 'Metric'])

# Assuming you have already split your data into x_train, x_test, y_train, and y_test

# Define the list of distance metrics to iterate over
distance_metrics = ['euclidean', 'chebyshev', 'minkowski','cosine']

for metric in distance_metrics:
    for k in range(1, 16):
        #knn = KNeighborsClassifier(n_neighbors=k, metric=metric)
        if metric == 'minkowski':
          knn = KNeighborsClassifier(n_neighbors=k, metric=metric, metric_params={'p': 3})
        else:
          knn = KNeighborsClassifier(n_neighbors=k, metric=metric)
        knn.fit(x_train, y_train)
        pred_i = knn.predict(x_test)

        # Calculate mean error and accuracy
        mean_error = np.mean(pred_i != y_test)
```

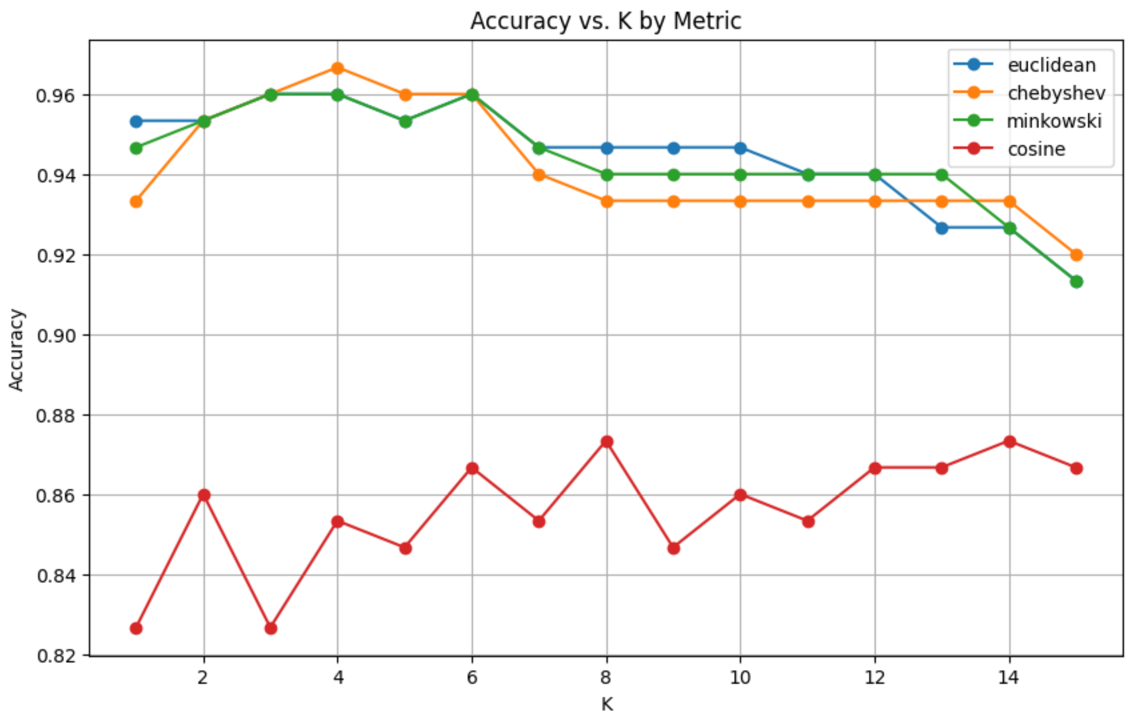
```
        acc = accuracy_score(y_test, pred_i)

        # Append values to the lists
        error.append(mean_error)
        accuracy.append(acc)
        metrics_list.append(metric)

        # Append values to the DataFrame
        results_df = results_df.append({'K': k, 'MeanError': mean_error, 'Accuracy': acc, 'Metric': metric}, ignore_index=Tru

results_df.to_csv("knn.csv", index=False)
```

```
import matplotlib.pyplot as plt
import pandas as pd

# Create a figure and axis
fig, ax = plt.subplots(figsize=(10, 6))
df=results_df

# Iterate over unique metrics to create lines
for metric in df['Metric'].unique():
    subset = df[df['Metric'] == metric]
    ax.plot(subset['K'], subset['MeanError'], marker='o', label=metric)

# Set labels and legend
ax.set_xlabel('K')
ax.set_ylabel('MeanError')
ax.set_title('MeanError vs. K by Metric')
ax.legend()

# Show the plot
plt.grid(True)
plt.show()
```




```
import matplotlib.pyplot as plt
import pandas as pd

# Create a figure and axis
fig, ax = plt.subplots(figsize=(10, 6))
df=results_df

# Iterate over unique metrics to create lines
for metric in df['Metric'].unique():
```

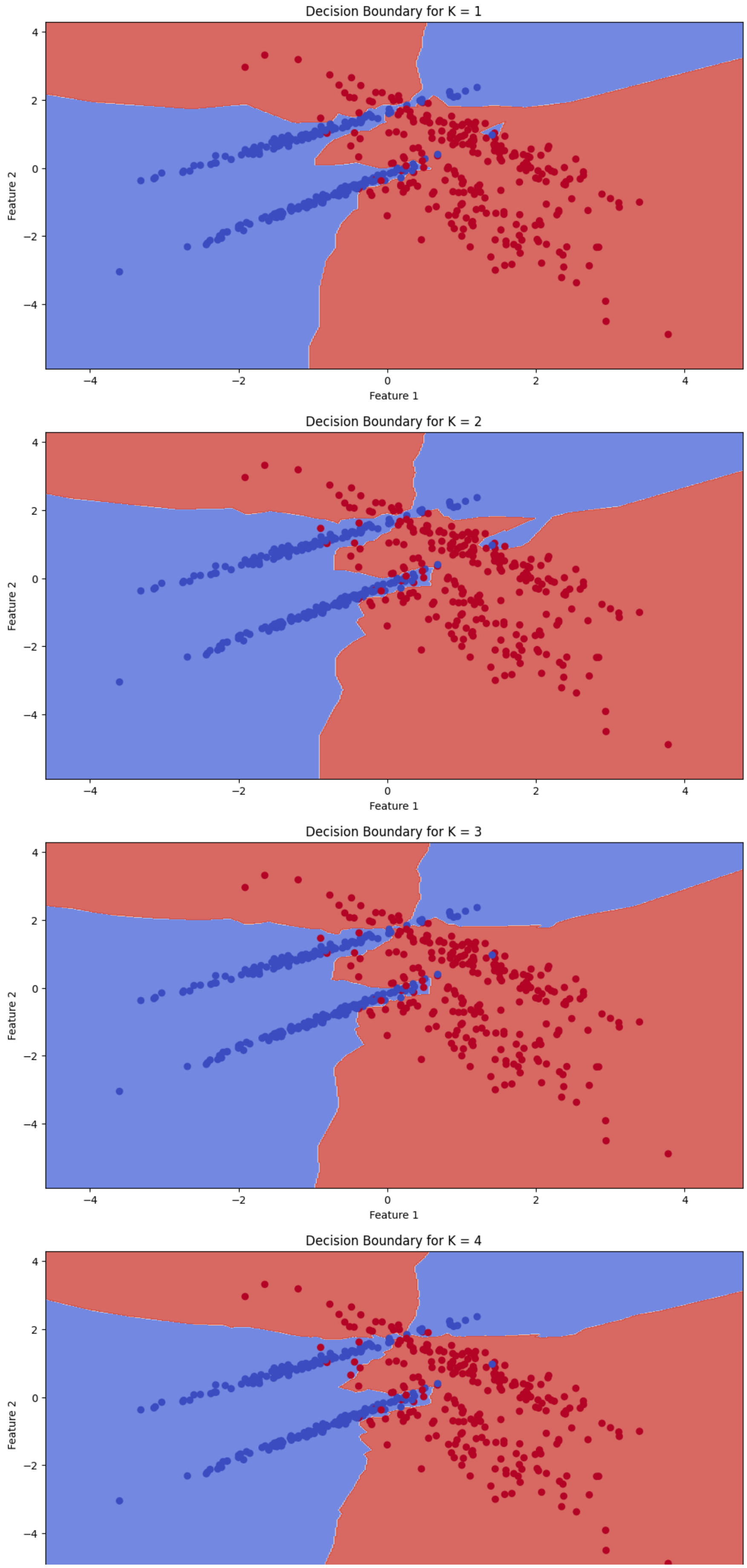
```
    subset = df[df['Metric'] == metric]
    ax.plot(subset['K'], subset['Accuracy'], marker='o', label=metric)

# Set labels and legend
ax.set_xlabel('K')
ax.set_ylabel('Accuracy')
ax.set_title('Accuracy vs. K by Metric')
ax.legend()

# Show the plot
plt.grid(True)
plt.show()
```



Create A synthetic dataset to demonstrate the Decision Boundaries at various Iterations

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
# Generate a synthetic dataset for visualization
X, y = make_classification(n_samples=500, n_features=2, n_informative=2, n_redundant=0, random_state=42)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Calculate error for K values between 1 and 30
error = []
for k in range(1, 6):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x_train, y_train)
    pred_i = knn.predict(x_test)
    error.append(np.mean(pred_i != y_test))

    # Create a meshgrid for the decision boundary plot
    h = 0.02  # Step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
    Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Plot the decision boundary for the current K
    plt.figure(figsize=(10, 5))
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
```

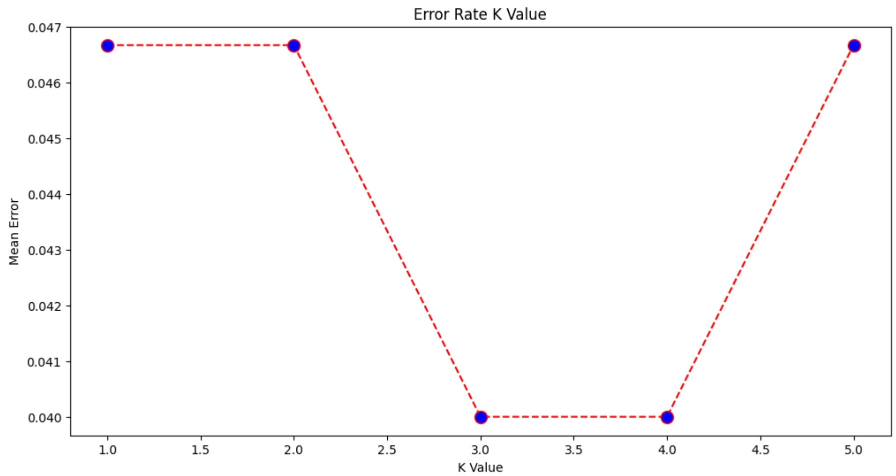
```
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
plt.title(f'Decision Boundary for K = {k}')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```




```

# Plot the error vs. K
plt.figure(figsize=(12, 6))
plt.plot(range(1, 6), error, color='red', linestyle='dashed', marker='o',
```

```
          markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
print("Minimum error:", min(error), "at K =", error.index(min(error)) + 1)
```

```
Minimum error: 0.04 at K = 3
```



Summary:

In this project we focused on evaluating the performance of a k-Nearest Neighbors (k-NN) classifier using different distance metrics. The primary objective is to determine the most suitable distance metric and the optimal value of k for a classification task. The code systematically explores a range of k values from 1 to 15 and four different distance metrics: Euclidean, Chebyshev, Minkowski with a specified parameter, and Cosine.

This exercise demonstrates the importance of hyperparameter tuning, specifically the choice of distance metric and the selection of k, in optimizing the performance of k-NN classifiers for classification tasks.

Also the resulting visualizations display the decision boundaries for different K values, helping to understand how the choice of K affects the model's capacity to distinguish between classes. This exercise serves as an informative illustration of the impact of K on k-NN classification performance.