Problem statement : Explain validation techiques. Implement anyone of them using the dataset and visualize the output.

Validation techniques are used in machine learning to assess the performance and generalization capabilities of a predictive model. They help us understand how well a model will perform on unseen data. One common validation technique is k-fold cross-validation. Here's an explanation of k-fold cross-validation and how to implement it using a dataset:

K-fold Cross-Validation:
K-fold cross-validation is a technique that partitions the dataset into 'k' equally sized subsets or folds. The model is trained and evaluated 'k' times, each time using a different fold as the validation set and the remaining folds as the training set. This helps ensure that the model is tested on different portions of the data, reducing the risk of overfitting or underfitting.

```
from sklearn import datasets
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

X, y = datasets.load_iris(return_X_y=True)

X = np.array(X)
y = np.array(y)
```

```
kf = KFold(n_splits=10, shuffle=True, random_state=42)

# Initialize a list to store accuracy scores
accuracy_scores = []

# Iterate through the folds
for k , ( train_idx, val_idx )in enumerate(kf.split(X)):
X_train, X_val = X[train_idx], X[val_idx]
y_train, y_val = y[train_idx], y[val_idx]

# Initialize and train a model (e.g., logistic regression)
model = LogisticRegression(solver='lbfgs', max_iter=1000,
C=0.01)
model.fit(X_train, y_train)

# Make predictions on the validation set
y_pred = model.predict(X_val)
```


```
kf = KFold(n_splits=10, shuffle=True, random_state=42)

# Initialize a list to store accuracy scores
accuracy_scores = []

# Iterate through the folds
for k , ( train_idx, val_idx )in enumerate(kf.split(X)):
X_train, X_val = X[train_idx], X[val_idx]
y_train, y_val = y[train_idx], y[val_idx]

# Initialize and train a model (e.g., logistic regression)
model = LogisticRegression(solver='lbfgs', max_iter=1000,
C=0.01)
model.fit(X_train, y_train)

# Make predictions on the validation set
y_pred = model.predict(X_val)
```

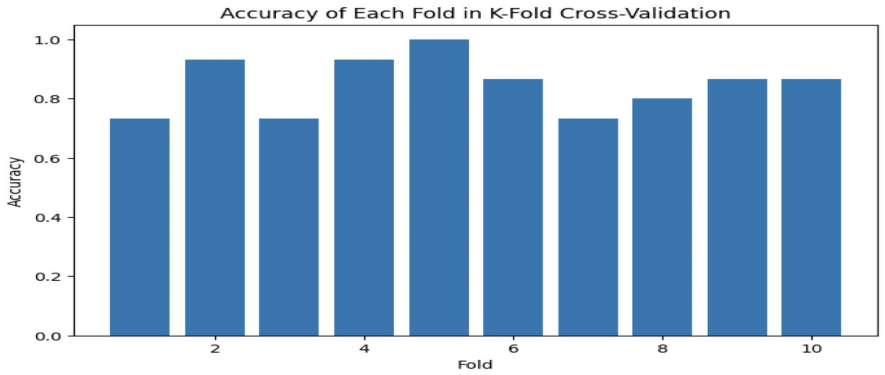
```python
# Calculate accuracy and store it
accuracy = accuracy_score(y_val, y_pred)
print(f'Fold : {k+1} Accuracy : {accuracy}%')
accuracy_scores.append(accuracy)

# Visualize the accuracy scores
plt.figure(figsize=(8, 5))
plt.bar(range(1, kf.get_n_splits() + 1), accuracy_scores)
plt.xlabel('Fold')
plt.ylabel('Accuracy')
plt.title('Accuracy of Each Fold in K-Fold Cross-Validation')
plt.show()

# Calculate and print the mean accuracy across all folds
mean_accuracy = np.mean(accuracy_scores)
print(f'Mean Accuracy: {mean_accuracy}')
```

Output :

```
Fold : 1 Accuracy : 0.7333333333333333%
Fold : 2 Accuracy : 0.9333333333333333%
Fold : 3 Accuracy : 0.7333333333333333%
Fold : 4 Accuracy : 0.9333333333333333%
Fold : 5 Accuracy : 1.0%
Fold : 6 Accuracy : 0.8666666666666667%
Fold : 7 Accuracy : 0.7333333333333333%
Fold : 8 Accuracy : 0.8%
Fold : 9 Accuracy : 0.8666666666666667%
Fold : 10 Accuracy : 0.8666666666666667%
```




```
Mean Accuracy: 0.8466666666666667
```