ECE 469/ECE 568 Machine Learning

Textbook:
Machine Learning: a Probabilistic Perspective by Kevin Patrick Murphy

Southern Illinois University

September 25, 2024

Implementation of K-Fold Cross Validation in Scikit-Learn

This lecture provides a sample code on implementing K-fold cross validation using Scikit-Learn.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Parameters
variance = 2
n = 1000  # Number of data points

# Generate data
x = np.random.uniform(-10, 10, n)
n = np.random.normal(0, np.sqrt(variance), x.shape)  # Gaussian
noise with mean 0 and variance
y = x ** 2 + 2 * x + 5 + n

# Create a DataFrame
data = pd.DataFrame({
'x': x,
'y': y
})
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Save the DataFrame to a CSV file
filename = f'dataset3_variance_{variance}.csv'
data.to_csv(filename, index=False)

print(f'Dataset saved to {filename}')

# Visualize the data
plt.figure(figsize=(8, 6))
plt.scatter(x, y, s=1, c='blue', alpha=0.5, label="Data points")
plt.title(f'Plot of x vs y (Variance = {variance})')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Load the dataset
filename = f'dataset3_variance_{variance}.csv'
data = pd.read_csv(filename)

# Extract x and y
x = data['x'].values
y = data['y'].values

# Reshape x (expects a 2D array for the features)
x = x.reshape(-1, 1)

# Parameters
max_degree = 5  # Maximum degree of the polynomial to test
k = 5  # Number of folds for cross-validation
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Initialize variables to track the best degree
best_degree = 1
best_mse = float('inf')
mse_per_degree = []
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Perform k-fold cross-validation for each polynomial degree
for degree in range(1, max_degree + 1):
kf = KFold(n_splits=k, shuffle=True, random_state=42)
mse_fold_values = []

# Initialize a plot for the regression fit
all_y_pred = np.zeros(x.shape)  # Ensure the
shape matches x (2D array)

# Loop through each fold
for train_index, test_index in kf.split(x):
x_train, x_test = x[train_index], x[test_index]
y_train, y_test = y[train_index], y[test_index]
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Transform the original x data into polynomial features
for the current degree
poly = PolynomialFeatures(degree=degree)
x_train_poly = poly.fit_transform(x_train)
x_test_poly = poly.transform(x_test)
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Initialize and fit the linear regression model
on the training data
poly_regressor = LinearRegression()
poly_regressor.fit(x_train_poly, y_train)

# Predict using the trained model on the test data
y_test_pred = poly_regressor.predict(x_test_poly)

# Calculate the mean squared error for the test set
mse_test = mean_squared_error(y_test, y_test_pred)
mse_fold_values.append(mse_test)

# Calculate the average MSE across all folds
for the current degree
avg_mse = np.mean(mse_fold_values)
mse_per_degree.append(avg_mse)
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Check if the current degree has the lowest average MSE
if avg_mse < best_mse:
best_mse = avg_mse
best_degree = degree


# Generate a new plot for this polynomial degree
plt.figure(figsize=(12, 8))
plt.scatter(x, y, s=10, color='blue', label='Original Data', ...
... alpha=0.6)

# Plot the polynomial fit for this degree
sorted_x = np.sort(x, axis=0)
sorted_x_poly = poly.transform(sorted_x)
plt.plot(sorted_x, poly_regressor.predict(sorted_x_poly),
 label=f'Degree {degree}', alpha=0.7)
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
# Check if the current degree has the lowest average MSE
if avg_mse < best_mse:
best_mse = avg_mse
best_degree = degree

# Generate a new plot for this polynomial degree
plt.figure(figsize=(12, 8))
plt.scatter(x, y, s=10, color='blue', label='Original Data',
alpha=0.6)

# Plot the polynomial fit for this degree
sorted_x = np.sort(x, axis=0)
sorted_x_poly = poly.transform(sorted_x)
plt.plot(sorted_x, poly_regressor.predict(sorted_x_poly),
label=f'Degree {degree}', alpha=0.7)
```

Implementation of K-Fold Cross Validation in Scikit-Learn

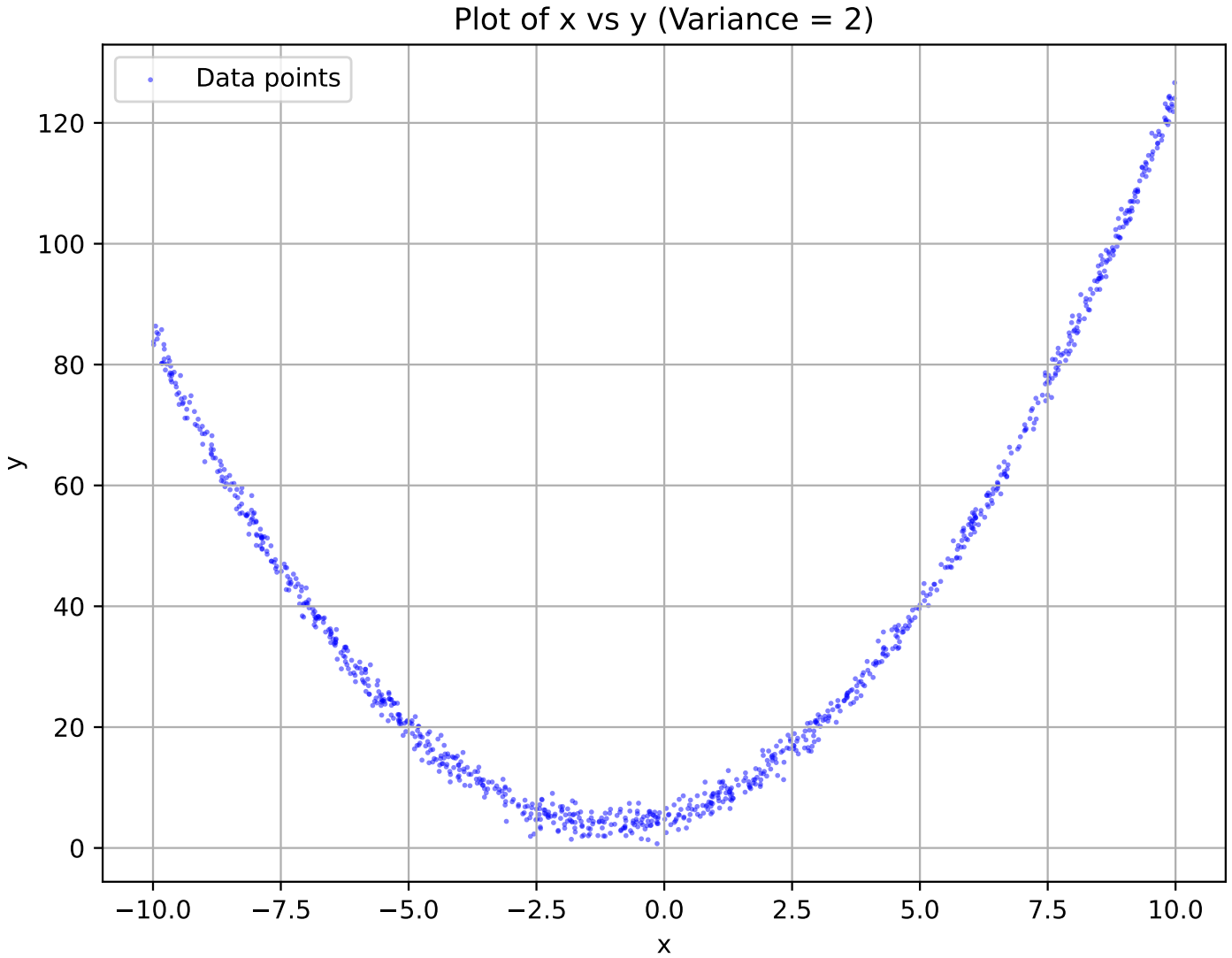
```
# Finalize the plot for this degree
plt.title(f'Polynomial Regression (Degree {degree}) - Variance = {v
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc='best')
plt.grid(True)
plt.show()


# Print the best polynomial degree and its corresponding MSE
for the current dataset
print(f"Results for variance = {variance}:")
print(f"  Best Polynomial Degree: {best_degree}")
```

Implementation of K-Fold Cross Validation in Scikit-Learn

```
plt.figure(figsize=(10, 6))
plt.plot(range(1, max_degree + 1), mse_per_degree, marker='o',
color='b', label='Cross-Validation MSE')
plt.title(f'MSE vs Polynomial Degree (Variance = {variance})')
plt.xlabel('Polynomial Degree')
plt.ylabel('Mean Squared Error (MSE)')
plt.xticks(range(1, max_degree + 1))
plt.legend()
plt.grid(True)
plt.show()
```

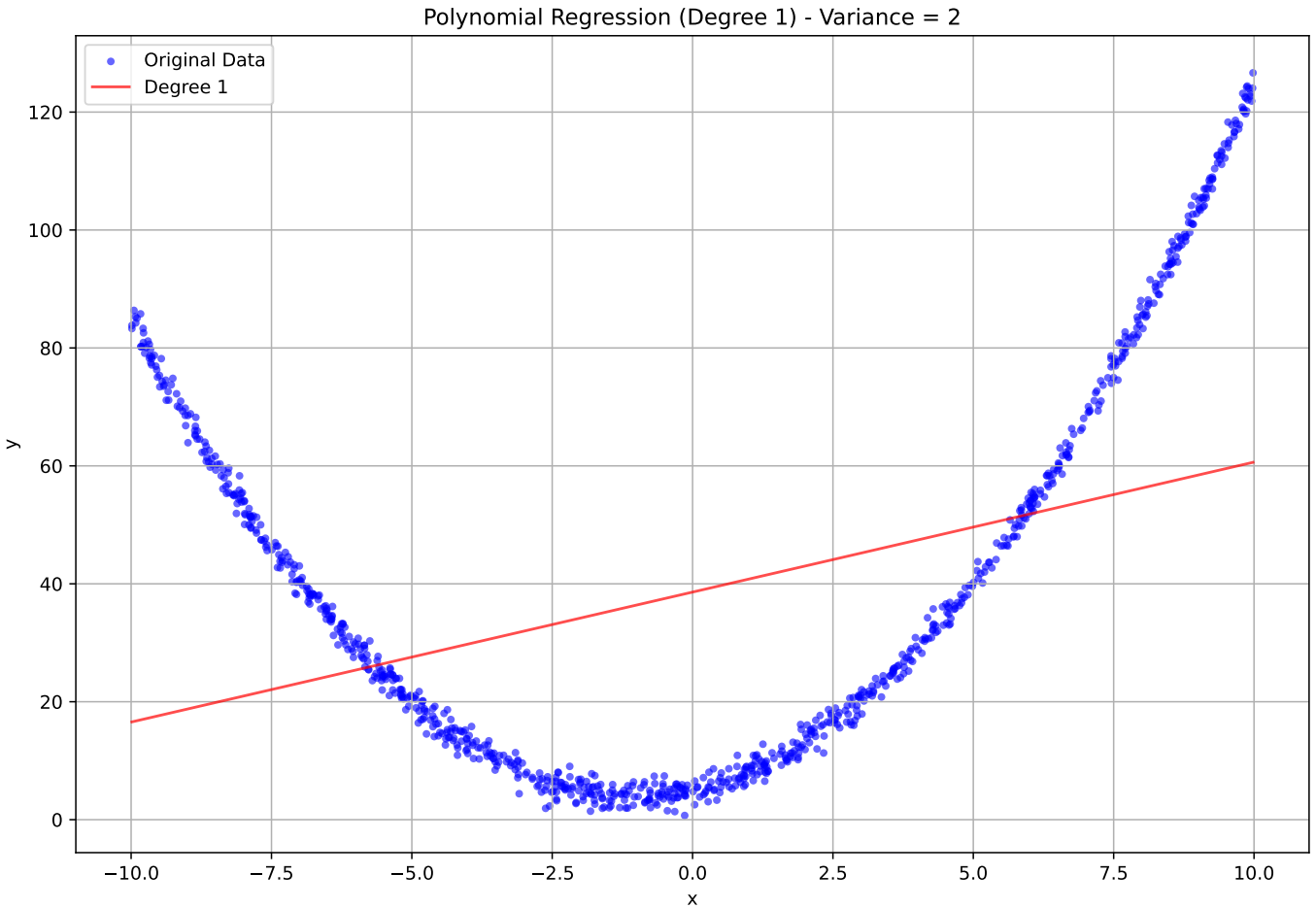
Implementation of K-Fold Cross Validation in Scikit-Learn



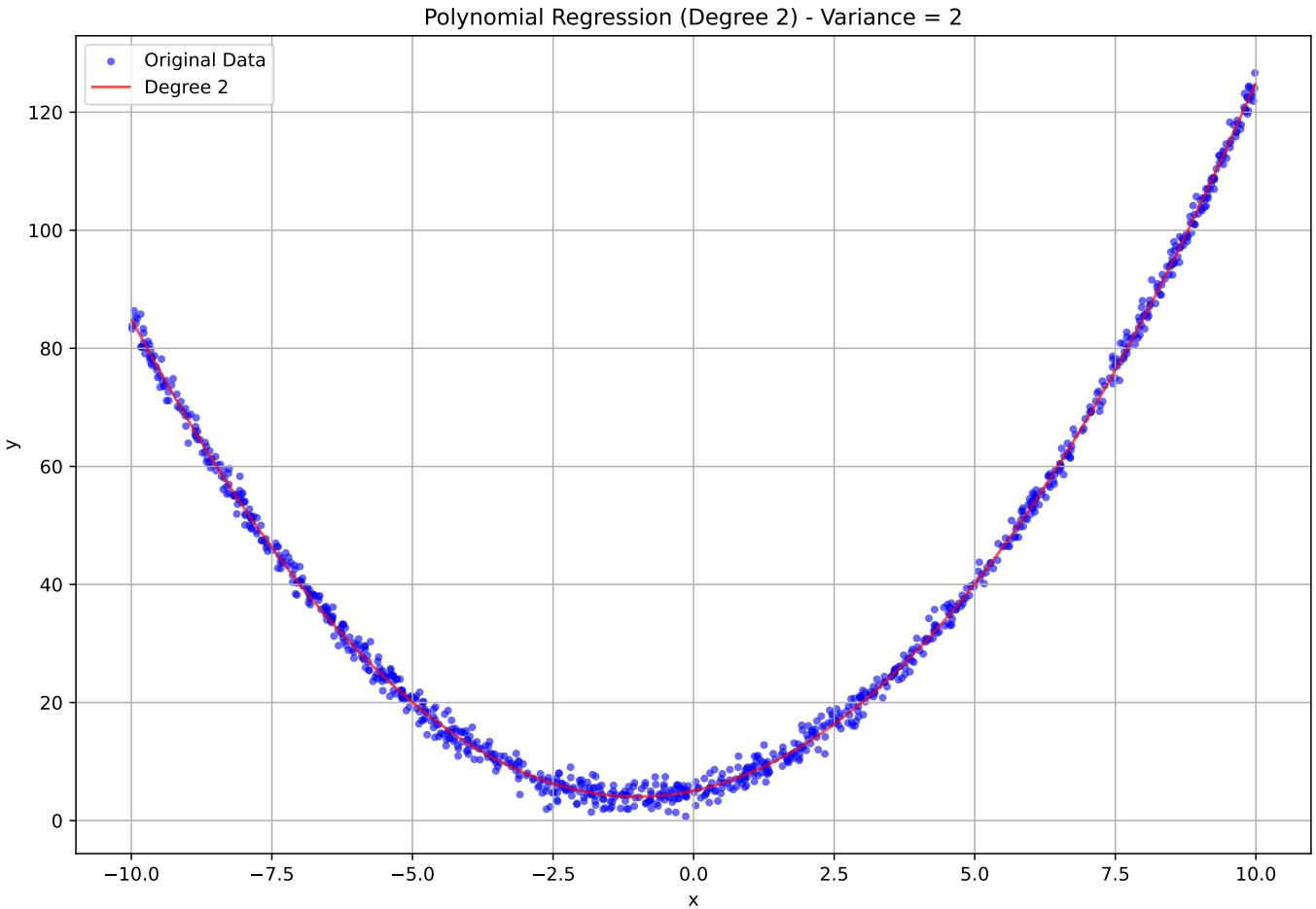
Implementation of K-Fold Cross Validation in Scikit-Learn

```
Dataset saved to dataset3_variance_2.csv
Results for variance = 2:
  Best Polynomial Degree: 2
  Best Average MSE across 5-folds: 1.9658288693763015
```

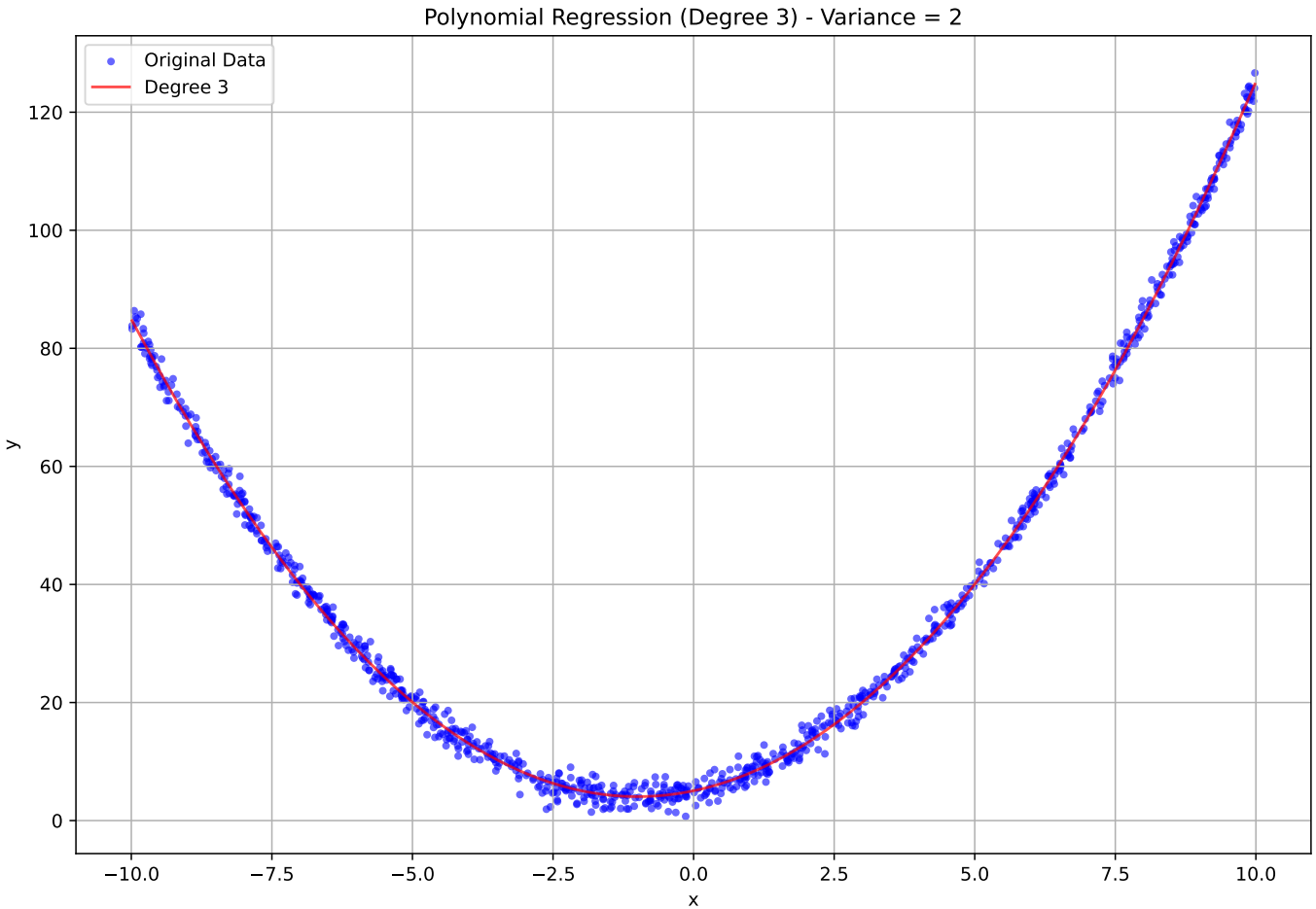
Implementation of K-Fold Cross Validation in Scikit-Learn



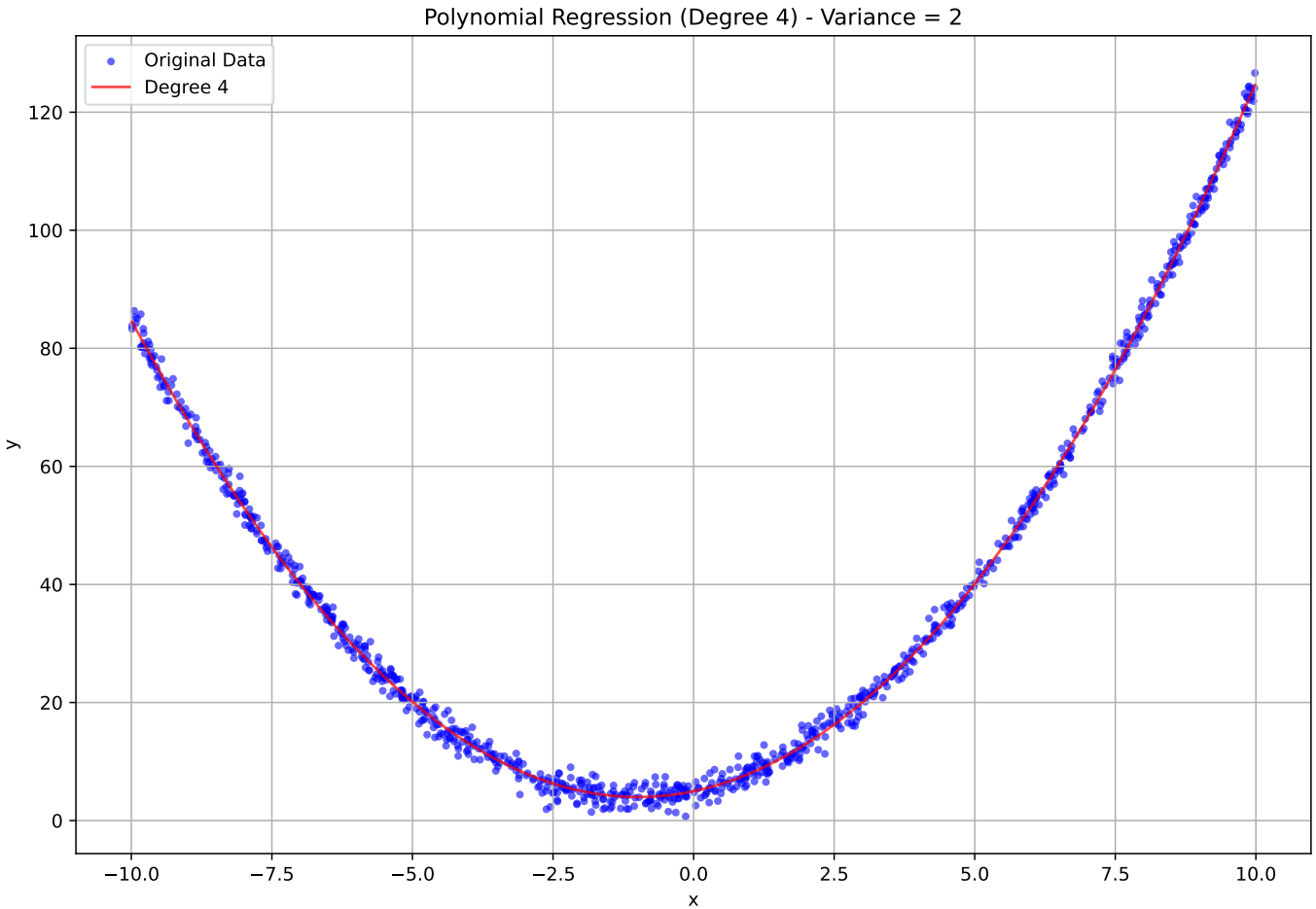
Implementation of K-Fold Cross Validation in Scikit-Learn



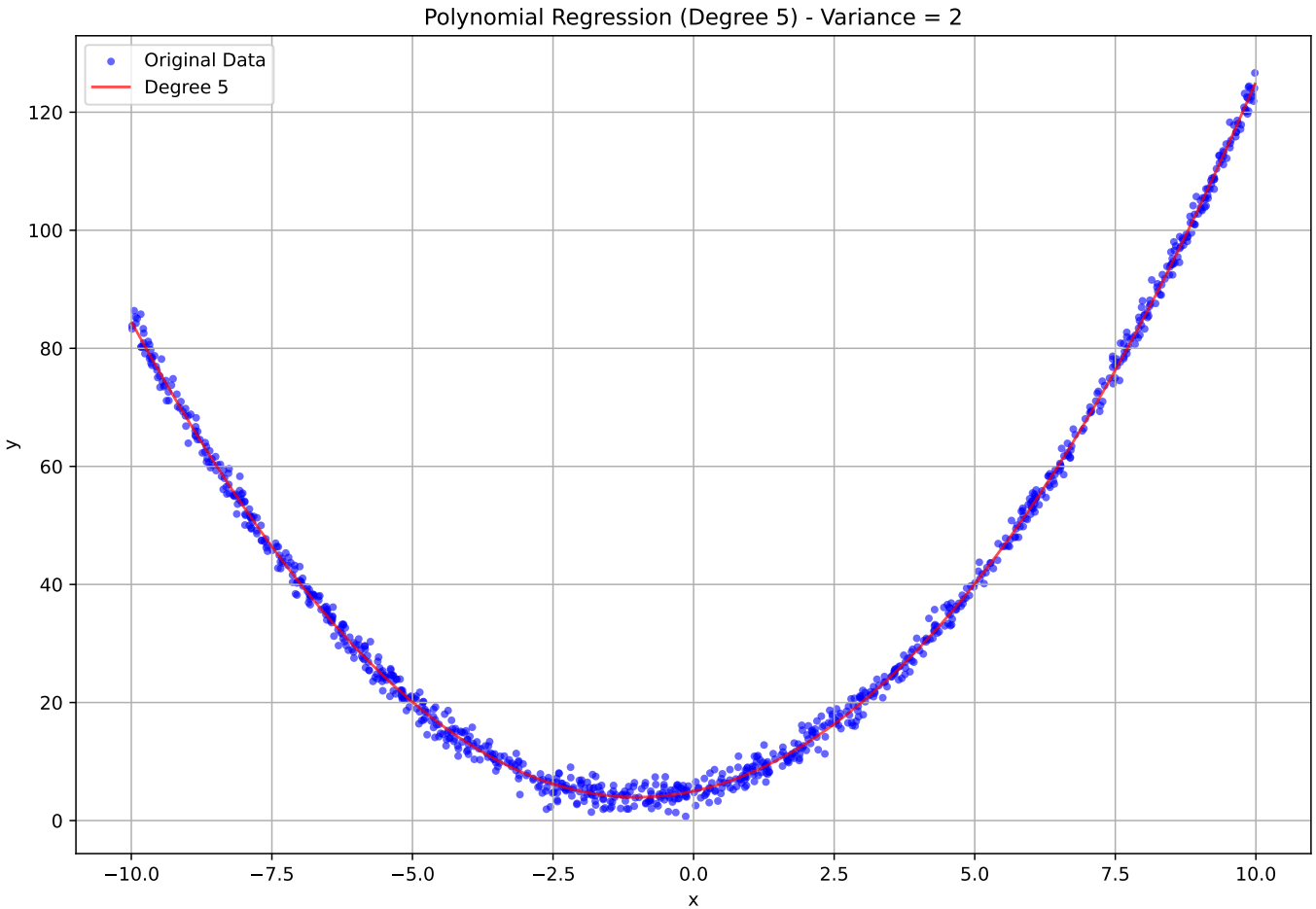
Implementation of K-Fold Cross Validation in Scikit-Learn



Implementation of K-Fold Cross Validation in Scikit-Learn



Implementation of K-Fold Cross Validation in Scikit-Learn



Implementation of K-Fold Cross Validation in Scikit-Learn