Stats 101C Homework 2

Damien Ha

In [1]:
```
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [2]:
```
np.random.seed(42)
n_samples = 1000
X = np.random.uniform(1, 2, (n_samples, 2))
```

In [3]:
```
# Define the functions
def function1(x):
    return x[:, 0]**2 + x[:, 1]**2 + x[:, 0]**3 + x[:, 1]**3

def function2(x):
    return (1 / x[:, 0]) + (1 / x[:, 1])
```

In [4]:
```
y1 = function1(X)
y2 = function2(X)
```

In [5]:
```
split_ratio = 0.7
n_train = int(n_samples * split_ratio)
X_train, X_test = X[:n_train], X[n_train:]
y1_train, y1_test = y1[:n_train], y1[n_train:]
y2_train, y2_test = y2[:n_train], y2[n_train:]
```

In [6]:
```
k_values = list(range(7, 42, 2))
results = []
```

In [7]:
```
for k in k_values:
    knn_model = KNeighborsRegressor(n_neighbors=k)
    linear_model = LinearRegression()

    knn_model.fit(X_train, y1_train)
```

```
    linear_model.fit(X_train, y1_train)

    knn_pred = knn_model.predict(X_test)
    linear_pred = linear_model.predict(X_test)

    mse_knn = mean_squared_error(y1_test, knn_pred)
    mse_linear = mean_squared_error(y1_test, linear_pred)

    results.append((k, mse_knn, mse_linear))
```

In [8]:

```
for k, mse_knn, mse_linear in results:
    print(f"KNN (k={k}) MSE: {mse_knn:.4f}, Linear Regression MSE: {mse_linear:.4f}")
```

```
KNN (k=7) MSE: 0.0485, Linear Regression MSE: 0.3782
KNN (k=9) MSE: 0.0498, Linear Regression MSE: 0.3782
KNN (k=11) MSE: 0.0543, Linear Regression MSE: 0.3782
KNN (k=13) MSE: 0.0626, Linear Regression MSE: 0.3782
KNN (k=15) MSE: 0.0696, Linear Regression MSE: 0.3782
KNN (k=17) MSE: 0.0747, Linear Regression MSE: 0.3782
KNN (k=19) MSE: 0.0807, Linear Regression MSE: 0.3782
KNN (k=21) MSE: 0.0844, Linear Regression MSE: 0.3782
KNN (k=23) MSE: 0.0943, Linear Regression MSE: 0.3782
KNN (k=25) MSE: 0.1009, Linear Regression MSE: 0.3782
KNN (k=27) MSE: 0.1118, Linear Regression MSE: 0.3782
KNN (k=29) MSE: 0.1249, Linear Regression MSE: 0.3782
KNN (k=31) MSE: 0.1328, Linear Regression MSE: 0.3782
KNN (k=33) MSE: 0.1415, Linear Regression MSE: 0.3782
KNN (k=35) MSE: 0.1525, Linear Regression MSE: 0.3782
KNN (k=37) MSE: 0.1605, Linear Regression MSE: 0.3782
KNN (k=39) MSE: 0.1707, Linear Regression MSE: 0.3782
KNN (k=41) MSE: 0.1796, Linear Regression MSE: 0.3782
```

In [9]:

```
results2 = []
for k in k_values:
    knn_model = KNeighborsRegressor(n_neighbors=k)
    linear_model = LinearRegression()

    knn_model.fit(X_train, y2_train)
    linear_model.fit(X_train, y2_train)

    knn_pred = knn_model.predict(X_test)
    linear_pred = linear_model.predict(X_test)
```

```
    mse_knn = mean_squared_error(y2_test, knn_pred)
    mse_linear = mean_squared_error(y2_test, linear_pred)

    results2.append((k, mse_knn, mse_linear))
```

In [10]:

```
for k, mse_knn, mse_linear in results2:
    print(f"KNN (k={k}) MSE: {mse_knn:.4f}, Linear Regression MSE: {mse_linear:.4f}")
```

```
KNN (k=7) MSE: 0.0001, Linear Regression MSE: 0.0013
KNN (k=9) MSE: 0.0001, Linear Regression MSE: 0.0013
KNN (k=11) MSE: 0.0001, Linear Regression MSE: 0.0013
KNN (k=13) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=15) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=17) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=19) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=21) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=23) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=25) MSE: 0.0002, Linear Regression MSE: 0.0013
KNN (k=27) MSE: 0.0003, Linear Regression MSE: 0.0013
KNN (k=29) MSE: 0.0003, Linear Regression MSE: 0.0013
KNN (k=31) MSE: 0.0003, Linear Regression MSE: 0.0013
KNN (k=33) MSE: 0.0003, Linear Regression MSE: 0.0013
KNN (k=35) MSE: 0.0003, Linear Regression MSE: 0.0013
KNN (k=37) MSE: 0.0004, Linear Regression MSE: 0.0013
KNN (k=39) MSE: 0.0004, Linear Regression MSE: 0.0013
KNN (k=41) MSE: 0.0004, Linear Regression MSE: 0.0013
```

For both functions, mean squared error for KNN is less than mean squared error for Linear Regression. So, in both cases, KNN outperforms Linear Regression