M6-L1 Problem 2

In this problem you'll learn how to make a 'pipeline' in SciKit-Learn. A pipeline chains together multiple sklearn modules and runs them in series. For example, you can create a pipeline to perform feature scaling and then regression. For more information see https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

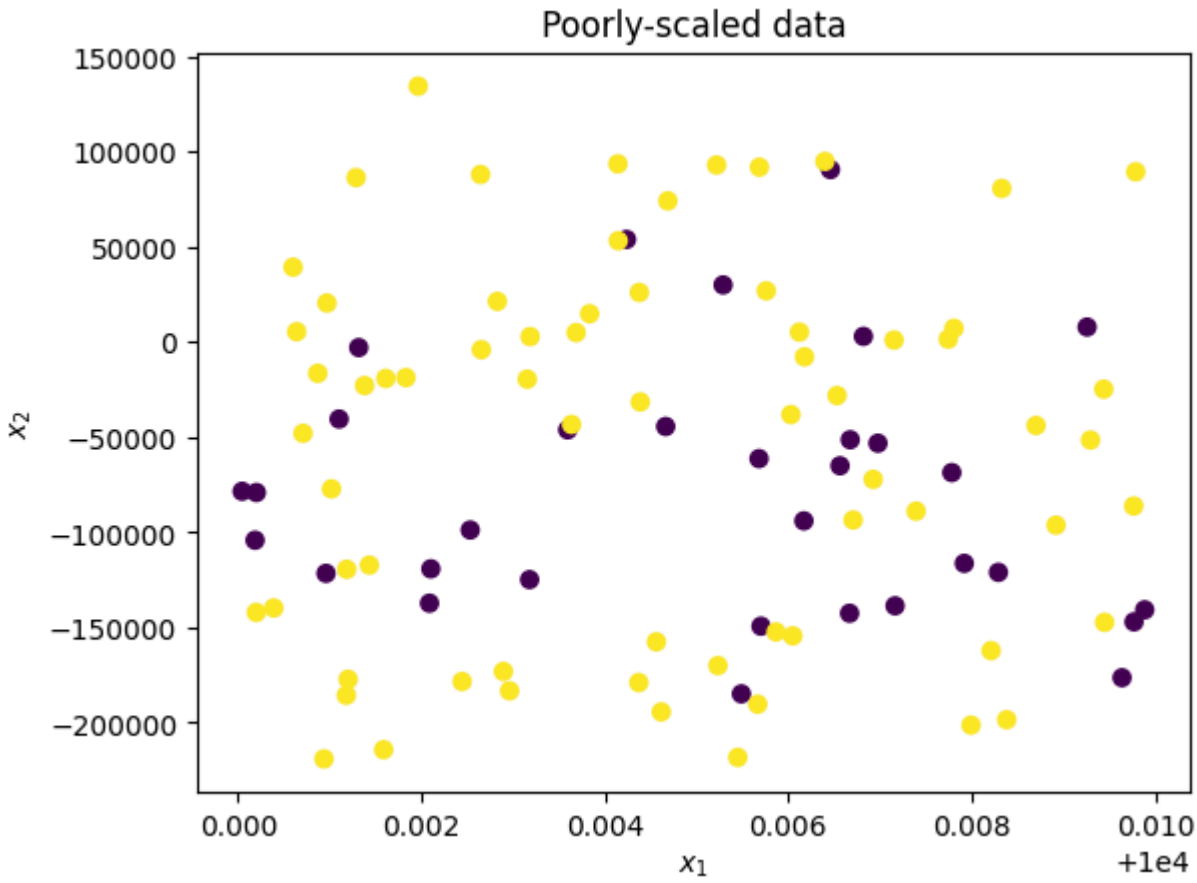
First, run the cell below to import modules and load data. Note the data axis scaling.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

x1 = np.array([10000.00548814, 10000.00715189, 10000.00602763, 10000.00544883, 10000.0
x2 = np.array([-184863.4856705 ,    1074.38382588,  -38090.38042426, -218261.93176495
X = np.vstack([x1,x2]).T
y = np.array([0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, train_size=0

plt.figure()
plt.scatter(x1,x2,c=y,cmap="viridis")
plt.xlabel("$x_1$")
plt.ylabel("$x_2$")
plt.title("Poorly-scaled data")
plt.show()
```



Creating a pipeline

In this section, code to set up a pipeline has been given. Make note of how each step works:

1. Create a scaler and classifier
2. Put the scaler and classifier into a new pipeline
3. Fit the pipeline to the training data
4. Make predictions with the pipeline

In [2]:
```
# Create a scaler and a classifier
scaler = MinMaxScaler()
model = KNeighborsClassifier()

# Put the scaler and classifier into a new pipeline
pipeline = Pipeline([("MinMax Scaler", scaler), ("KNN Classifier", model)])

# Fit the pipeline to the training data
pipeline.fit(X_train, y_train)

# Make predictions with the pipeline
pred_train = pipeline.predict(X_train)
pred_test = pipeline.predict(X_test)
print("Training accuracy:", accuracy_score(y_train, pred_train), "   Testing accuracy:
```

```
Training accuracy: 0.825    Testing accuracy: 0.6
```

Testing several pipelines

Now, complete the code to create a new pipeline for every combination of scalers and models below:

Scalers:

- None
- MinMax
- Standard

Classifiers:

- Logistic Regression
- Support Vector Machine
- KNN Classifier, 1 neighbor

Within the loop, a scaler and model are created. You will create a pipeline, fit it to the training data, and make predictions on testing and training data.

In [3]:

```
def get_scaler(i):
    if i == 0:
        return ("No Scaler", None)
    elif i == 1:
        return ("MinMax Scaler", MinMaxScaler())
    elif i == 2:
        return ("Standard Scaler", StandardScaler())

def get_model(i):
    if i == 0:
        return ("Logistic Regression", LogisticRegression())
    elif i == 1:
        return ("Support Vector Classifier", SVC())
    elif i == 2:
        return ("1-NN Classifier", KNeighborsClassifier(n_neighbors=1))

for scaler_index in range(3):
    for model_index in range(3):
        scaler = get_scaler(scaler_index)
        model = get_model(model_index)

        # YOUR CODE GOES HERE
        # Create a pipeline
        # Fit the pipeline on X_train, y_train
        # Calculate acc_train and acc_test for the pipeline

        pipeline = Pipeline([scaler, model])
        # Fit the pipeline to the training data
        pipeline.fit(X_train, y_train)
        pred_train = pipeline.predict(X_train)
        pred_test = pipeline.predict(X_test)

        acc_train =  np.sum(pred_train == y_train) / len(y_train)
        acc_test =  np.sum(pred_test == y_test) / len(y_test)

        print(f"{scaler[0]:>15},{model[0]:>26}:    Train Acc. = {100*acc_train:5.1f}%
```

```
      No Scaler,       Logistic Regression:    Train Acc. =  67.5%    Test Acc. =  70.
0%
      No Scaler, Support Vector Classifier:    Train Acc. =  78.8%    Test Acc. =  65.
0%
      No Scaler,           1-NN Classifier:    Train Acc. = 100.0%    Test Acc. =  50.
0%
  MinMax Scaler,       Logistic Regression:    Train Acc. =  67.5%    Test Acc. =  70.
```

```
0%
  MinMax Scaler, Support Vector Classifier:    Train Acc. =  67.5%    Test Acc. =  70.
0%
  MinMax Scaler,           1-NN Classifier:    Train Acc. = 100.0%    Test Acc. =  85.
0%
Standard Scaler,       Logistic Regression:    Train Acc. =  67.5%    Test Acc. =  70.
0%
Standard Scaler, Support Vector Classifier:    Train Acc. =  68.8%    Test Acc. =  70.
0%
Standard Scaler,           1-NN Classifier:    Train Acc. = 100.0%    Test Acc. =  85.
0%
```

Questions

Answer the following questions:

1. Which model's testing accuracy was improved the most by scaling data?

1. Which performs better on this data: MinMax scaler, Standard scaler, or neither?

1. Support vector model was imporved the most
2. Standard scaler perform slightly better than MinMax Scaler, but still not great.