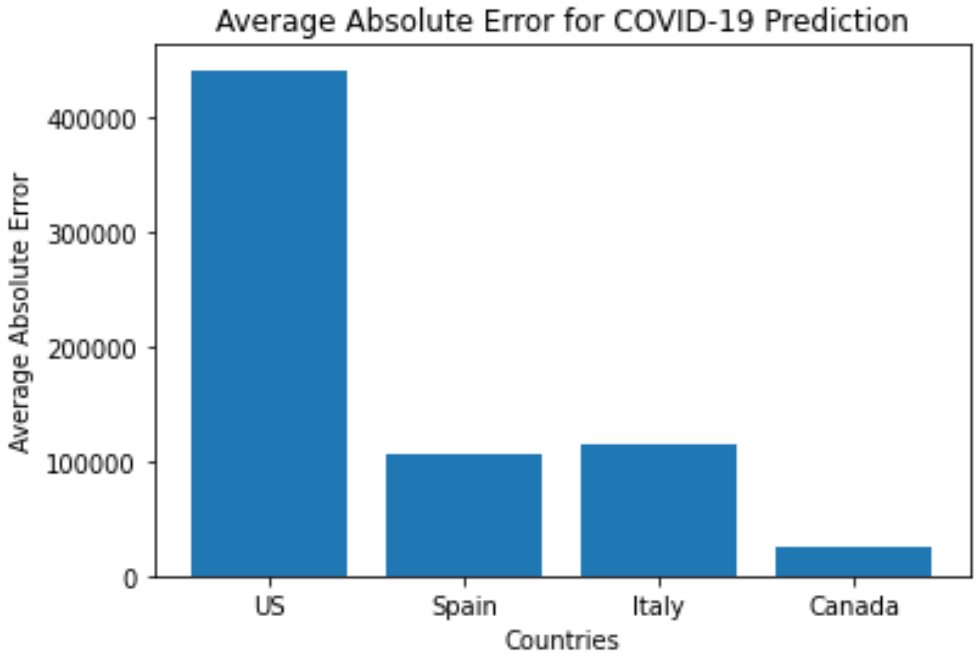
Shreeya Sampat

Covid_Data: Neural Networks



Performance of Cross Validation:

Setting	R^2 Score
S1	0.9257
S2	0.8697
S3	0.9150
S4	0.9317
S5	0.9381
S6	0.9396
S7	0.9072
S8	0.9248

According to the R^2 scores, the best-performing setting is S6. S6 contains three hidden layers with 10 neurons in each layer. This setting shows the highest average performance across the folds during cross-validation. Hence, this setting can now be used to predict new cases of COVID-19.

Python Code:

```
import csv
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.neural_network import MLPRegressor
import matplotlib.pyplot as plt

# Function to load data
def load_data(filename):
```

```
    glob = dict()
    with open(filename, newline='') as csvfile:
        datareader = csv.reader(csvfile)
        for r in datareader:
            c = r[0]
            if c == 'week':
                weekID = r[1:]
            else:
                tmp = [0] * 21
                darray = r[1:]
                for i in range(len(darray)):
                    t = int(weekID[i])
                    d = int(darray[i])
                    if t < 21:
                        tmp[t] += d
                glob[c] = tmp
    return glob

# Function to calculate new cases
def calculate_new_cases(glob):
    for c in glob:
        tmp = glob[c]
        tmp2 = [tmp[0]]
        for i in range(1, len(tmp)):
            tmp2.append(tmp[i] - tmp[i-1])
        glob[c] = tmp2
    return glob

# Function to prepare dataset
def prepare_dataset(glob):
    X = []
    Y = []
    step = 10
    for c in glob:
        tmp = glob[c]
        for j in range(len(tmp)-step-1):
            stest = sum(tmp[j: j + step])
            if stest > 0:
                X.append(tmp[j: j + step])
                Y.append(tmp[j + step])
    return np.array(X), np.array(Y)

# Load data
filename = 'covid19_global_dataset.csv'
glob = load_data(filename)

# Calculate new cases
glob = calculate_new_cases(glob)

# Prepare dataset
```

Covid_Data: Neural Networks

```
X, Y = prepare_dataset(glob)

# Select countries for test
countries = ['US', 'Spain', 'Italy', 'Canada']
X_test = np.array([glob[c][-10:] for c in countries])

# Define problem settings
settings = [
    (15, 15), (10, 10), (10, 15), (15, 10),
    (15, 15, 15), (10, 10, 10),
    (10,), (15,)
]

# Perform cross-validation and compare settings
scores = []
for setting in settings:
    clf = MLPRegressor(hidden_layer_sizes = setting, random_state = 1, max_iter = 2000)
    kfold = KFold(n_splits = 5, shuffle = True, random_state = 1)
    cv_scores = cross_val_score(clf, X, Y, cv = kfold, scoring = 'r2')
    scores.append(cv_scores.mean())

best_setting_index = np.argmax(scores)
best_setting = settings[best_setting_index]

# Train model with best setting
best_clf = MLPRegressor(hidden_layer_sizes=best_setting, random_state=1, max_iter=2000)
best_clf.fit(X, Y)

# Predict new cases for test countries
predictions = best_clf.predict(X_test)

# Calculate average absolute error
avg_absolute_errors = []
for i, country in enumerate(countries):
    avg_absolute_error = np.mean(np.abs(predictions[i] - glob[country][-10:]))
    avg_absolute_errors.append(avg_absolute_error)

# Plot bar chart
plt.bar(countries, avg_absolute_errors)
plt.xlabel('Countries')
plt.ylabel('Average Absolute Error')
plt.title('Average Absolute Error for COVID-19 Prediction')
plt.show()

# Report performances in table
print("Performance of Cross Validation:")
print("Setting\t\tR2 Score")
for i, score in enumerate(scores):
    print(f"S{i+1}\t\t{score:.4f}")
```