In [ ]:

```
import pandas as pd

file_path = '/content/Valhalla23.csv'
data = pd.read_csv(file_path)

data.head()
```

Out[ ]:

	Celsius	Valks
0	61.4720	-139.740
1	70.5790	-156.600
2	-7.3013	73.269
3	71.3380	-165.420
4	43.2360	-75.835

In [ ]:

In [ ]:

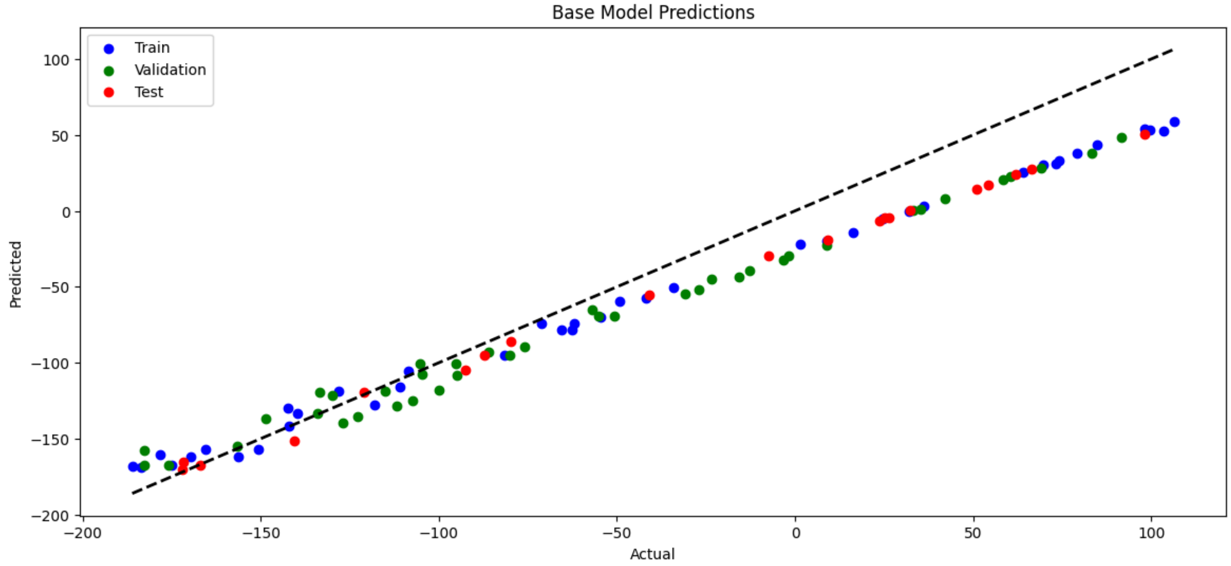
In [ ]:

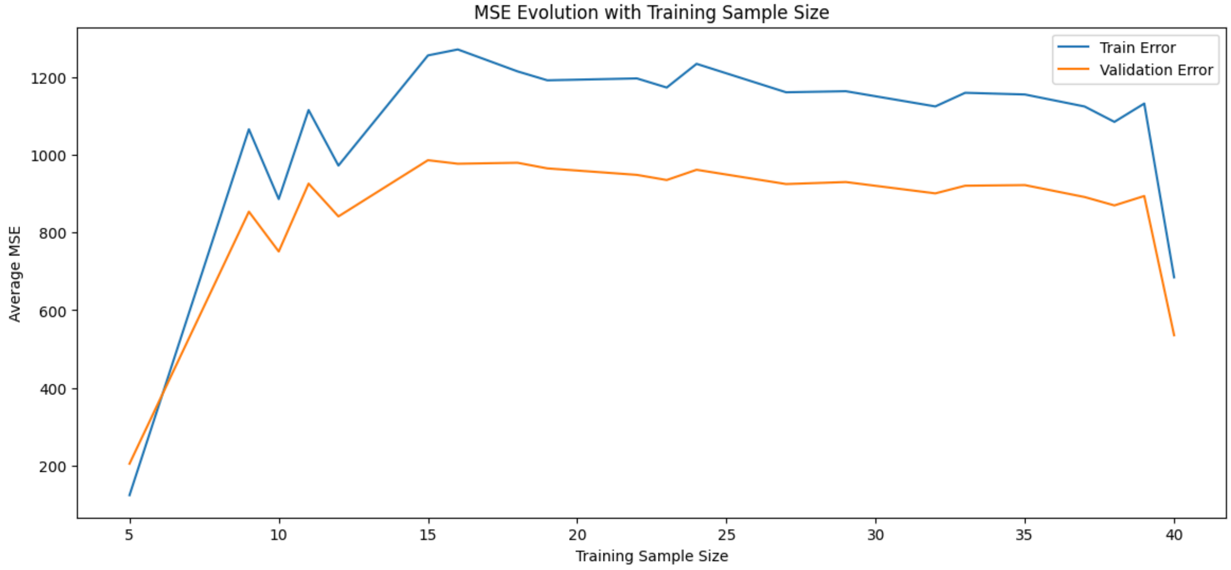
In [ ]:

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDRegressor
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Seed
seed = 1214

# Load dataset
file_path = '/content/Valhalla23.csv'
data = pd.read_csv(file_path)

X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Split the data into training (40%), validation (40%), and test (20%) sets
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.6, random_state=
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.33, random

# Train using SGDRegressor
base_model = SGDRegressor(learning_rate='constant', eta0=1E-4, max_iter=1000000, rando
base_model.fit(X_train, y_train)

# Calculate MSE for training, validation, and test sets
mse_train_base = mean_squared_error(y_train, base_model.predict(X_train))
mse_val_base = mean_squared_error(y_val, base_model.predict(X_val))
mse_test_base = mean_squared_error(y_test, base_model.predict(X_test))
```

```
# Generate predictions for plotting
y_train_pred = base_model.predict(X_train)
y_val_pred = base_model.predict(X_val)
y_test_pred = base_model.predict(X_test)

# Plot the results
plt.figure(figsize=(14, 6))
plt.scatter(y_train, y_train_pred, color='blue', label='Train')
plt.scatter(y_val, y_val_pred, color='green', label='Validation')
plt.scatter(y_test, y_test_pred, color='red', label='Test')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)
plt.xlabel('Actual')
plt.ylabel('Predicted')
plt.title('Base Model Predictions')
plt.legend()
plt.show()

# Create a list with 20 elements between 2 and 39, without repetition
sample_sizes = sorted(np.random.choice(range(2, 40), 20, replace=False))

avg_mse_train = []
avg_mse_val = []

# Train 100 models
for size in sample_sizes:
    mse_train_list = []
    mse_val_list = []
    for _ in range(100):
        X_train_sample, _, y_train_sample, _ = train_test_split(X_train, y_train, trai
        model = SGDRegressor(learning_rate='constant', eta0=1E-4, max_iter=1000000, ra
        model.fit(X_train_sample, y_train_sample)
        mse_train_list.append(mean_squared_error(y_train_sample, model.predict(X_train
        mse_val_list.append(mean_squared_error(y_val, model.predict(X_val)))
    avg_mse_train.append(np.mean(mse_train_list))
    avg_mse_val.append(np.mean(mse_val_list))

avg_mse_train.append(mse_train_base)
avg_mse_val.append(mse_val_base)

# Plot the evolution of the average MSE
plt.figure(figsize=(14, 6))
plt.plot(sample_sizes + [len(X_train)], avg_mse_train, label='Train Error')
plt.plot(sample_sizes + [len(X_train)], avg_mse_val, label='Validation Error')
plt.xlabel('Training Sample Size')
plt.ylabel('Average MSE')
plt.title('MSE Evolution with Training Sample Size')
plt.legend()
plt.show()

mse_train_base, mse_val_base, mse_test_base, avg_mse_train[-2:], avg_mse_val[-2:]
```





Out[ ]:

```
(684.6345282622414,
 535.5997921787242,
 685.213508707932,
 [1132.0536988954248, 684.6345282622414],
 [894.0572370597533, 535.5997921787242])
```

In [ ]:

```
%%shell
jupyter nbconvert --to html /content/Port_Imple_WFRAMEWORK_A01571214_Lautaro_Coteja.ip
```