Out Of Sample Learning

Approach: I perform regression for three functions, x^2, x^2 + y^2 and random function in 6 dimensions using three tools, neural networks, SVR, KNN regression.

Function 1 =X^2, from [-1,1] values from [-0.5,0.5] are masked and model should be able to predict that. The red region in the curve is the masked region.

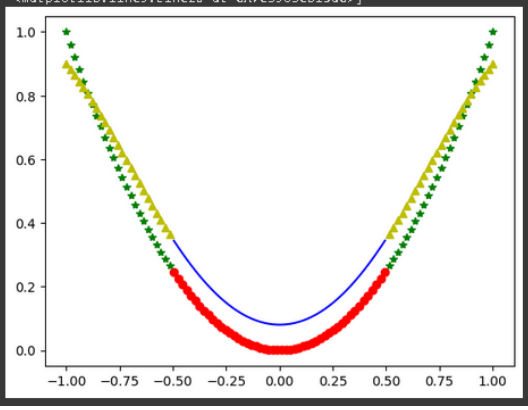
Using SVR: MSR was just 0.007.

```
[ ]  1 from sklearn.svm import SVR
     2 from sklearn.neural_network import MLPRegressor
     3 from sklearn.neighbors import KNeighborsRegressor
     4 from sklearn.metrics import mean_squared_error
     5 svr = SVR(kernel='rbf')
     6 svr.fit(X1_train.reshape(-1,1), y1_train)
     7 y1_svm_pred = svr.predict((X1_test).reshape(-1,1))
     8 y1_svm_train_pred = svr.predict((X1_train).reshape(-1,1))
     9 mse_svm = mean_squared_error(y1_test, y1_svm_pred)
    10 print(f"Mean Squared Error for SVM: {mse_svm}")
    11
```

```
Mean Squared Error for SVM: 0.007744928438523044
```

```
[ ]  1 plt.plot(X1_test,y1_test,'ro',X1_test,y1_svm_pred,'b-',X1_train,y1_train,'g*',X1_train,y1_svm_train_pred,'y^')
```

```
[<matplotlib.lines.Line2D at 0x7c3983cb1330>,
 <matplotlib.lines.Line2D at 0x7c3983cb13f0>,
 <matplotlib.lines.Line2D at 0x7c3983cb13c0>,
 <matplotlib.lines.Line2D at 0x7c3983cb15a0>]
```



Using Neural Network: MSR was just 0.0004. Two layers of 100 & 50 neurons were used.

```
[ ]  1 nn = MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000)
     2 nn.fit(X1_train.reshape(-1,1), y1_train)
     3
```

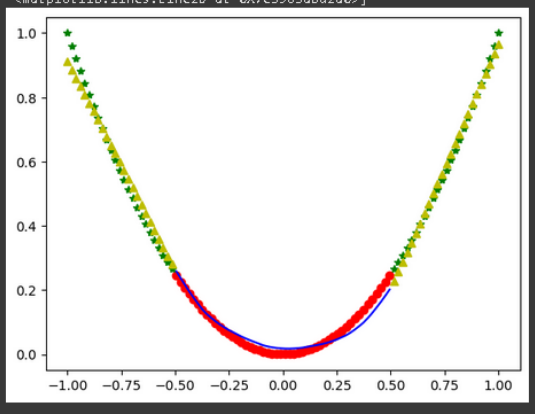
```
MLPRegressor
MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000)
```

```
[ ]  1 y1_nn_pred = nn.predict(X1_test.reshape(-1, 1))
     2 y1_nn_train_pred = nn.predict(X1_train.reshape(-1, 1))
     3 mse_nn = mean_squared_error(y1_test, y1_nn_pred)
     4 mse_nn
```

```
0.0004923225740392799
```

```
[ ]  1 plt.plot(X1_test,y1_test,'ro',X1_test,y1_nn_pred,'b-',X1_train,y1_train,'g*',X1_train,y1_nn_train_pred,'y^')
```

```
[<matplotlib.lines.Line2D at 0x7c3983abc190>,
 <matplotlib.lines.Line2D at 0x7c3983abd0f0>,
 <matplotlib.lines.Line2D at 0x7c3983abc0a0>,
 <matplotlib.lines.Line2D at 0x7c3983abd2a0>]
```



Using KNN regressor: Hyperparameter set was 20 neighbors. High MSR of 0.155.

```
1 knn = KNeighborsRegressor(n_neighbors=20)
2 knn.fit(X1_train.reshape(-1, 1) , y1_train)
```

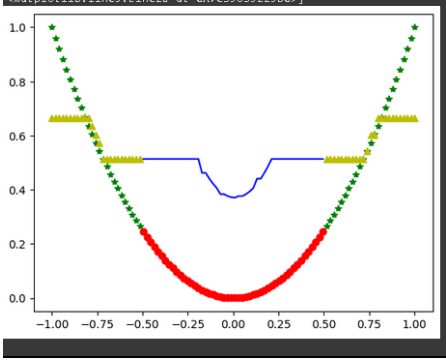
```
KNeighborsRegressor
KNeighborsRegressor(n_neighbors=20)
```

```
1 y1_knn_pred = knn.predict(X1_test.reshape(-1,1))
2 y1_knn_pred_train = knn.predict(X1_train.reshape(-1,1))
3 mse_knn = mean_squared_error(y1_test, y1_knn_pred)
4 mse_knn
```

```
0.15514528068880906
```

```
1 plt.plot(X1_test,y1_test,'ro',X1_test,y1_knn_pred,'b-',X1_train,y1_train,'g*',X1_train,y1_knn_pred_train,'y^')
```

```
[<matplotlib.lines.Line2D at 0x7c3983922740>,
 <matplotlib.lines.Line2D at 0x7c3983922800>,
 <matplotlib.lines.Line2D at 0x7c39839227d0>,
 <matplotlib.lines.Line2D at 0x7c39839229b0>]
```



Function 2= X^2 + Y^2 running from X [-0.5,0.5] and Y [-0.5,0.5].

SVR: MSR was 0.0005.

```
[ ]  1 from sklearn.svm import SVR
     2 from sklearn.neural_network import MLPRegressor
     3 from sklearn.neighbors import KNeighborsRegressor
     4
     5
     6 svr = SVR(kernel='rbf')
     7 svr.fit(X_train, y_train)
     8
     9
    10
```

```
▾ SVR
SVR()
```

```
[ ]  1 from sklearn.metrics import mean_squared_error
     2 y_svm_pred = svr.predict(X_test)
     3 y_svm_pred_train = svr.predict(X_train)
     4 mse_svm = mean_squared_error(y_test, y_svm_pred)
     5 mse_svm
```

```
0.0005073552187271591
```

Neural Network: 4 Hidden layers were used . Increasing neurons and layers was decreasing MSR . For this case MSR was 0.010.

```
12 mask = (X2[:, 0] < -0.5) | (X2[:, 0] > 0.5) | (X2[:, 1] < -0.5) | (X2[:, 1] > 0.5)
13 X_train, y_train = X2[mask], y2[mask]
14 X_test, y_test = X2[~mask], y2[~mask]
15
16
17 nn = MLPRegressor(hidden_layer_sizes=(300, 200,100,50), max_iter=1000)
18 nn.fit(X_train, y_train)
19
20
21 y_nn_pred = nn.predict(X_test)
22
23
24 mse_nn = mean_squared_error(y_test, y_nn_pred)
25 print(mse_nn)
26 fig = plt.figure()
27 ax = fig.add_subplot(111, projection='3d')
28
29 ax.scatter(X_test[:, 0], X_test[:, 1], y_test, c='red', marker='o', label='Test Data')
30 ax.scatter(X_train[:, 0], X_train[:, 1], y_train, c='green', marker='*', label='Training Data')
31 ax.scatter(X_test[:, 0], X_test[:, 1], y_nn_pred, c='blue', marker='^', label='SVR Predictions (Test)')
32
33 ax.set_xlabel('X-axis')
34 ax.set_ylabel('Y-axis')
35 ax.set_zlabel('Y-value')
36 ax.set_title('3D Scatter Plot of NN Regression Results')
37 ax.legend()
38 plt.show()
39 from mpl_toolkits.mplot3d import Axes3D
40 fig = plt.figure()
41 ax = fig.add_subplot(111, projection='3d')
42
43 x_grid, y_grid = np.meshgrid(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100))
44 X_grid = np.column_stack((x_grid.ravel(), y_grid.ravel()))
45
46 y_grid_pred = nn.predict(X_grid)
47
48 ax.plot_surface(x_grid, y_grid, y_grid_pred.reshape(x_grid.shape), cmap='viridis')
49
50 ax.set_xlabel('X-axis')
51 ax.set_ylabel('Y-axis')
52 ax.set_zlabel('Y-value')
53 ax.set_title('3D Surface Plot of NN Regression Results')
54 plt.show()
55
```

```
0.010373118607606854
```

KNN regressor: MSR was 0.08. Neighbours used were only 5 here.

```python
mask = (X2[:, 0] < -0.5) | (X2[:, 0] > 0.5) | (X2[:, 1] < -0.5) | (X2[:, 1] > 0.5)
X_train, y_train = X2[mask], y2[mask]
X_test, y_test = X2[~mask], y2[~mask]

knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)


y_knn_pred = knn.predict(X_test)


mse_knn = mean_squared_error(y_test, y_knn_pred)
print(mse_knn)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X_test[:, 0], X_test[:, 1], y_test, c='red', marker='o', label='Test Data')
ax.scatter(X_train[:, 0], X_train[:, 1], y_train, c='green', marker='*', label='Training Data')
ax.scatter(X_test[:, 0], X_test[:, 1], y_svm_pred, c='blue', marker='^', label='SVR Predictions (Test)')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Y-value')
ax.set_title('3D Scatter Plot of KNN Regression Results')
ax.legend()
plt.show()
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x_grid, y_grid = np.meshgrid(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100))
X_grid = np.column_stack((x_grid.ravel(), y_grid.ravel()))

y_grid_pred = nn.predict(X_grid)

ax.plot_surface(x_grid, y_grid, y_grid_pred.reshape(x_grid.shape), cmap='viridis')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Y-value')
ax.set_title('3D Surface Plot of KNN Regression Results')
plt.show()

```

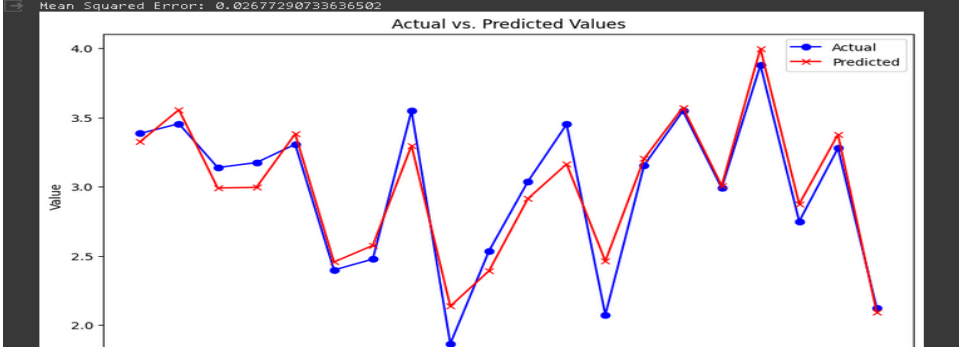
```
0.08184118274653249
```

Function 3: Sum of 6 Random variables.

Using SVR: MSR for 0.026

```python
X_train, X_test, y_train, y_test = train_test_split(X6, y6, test_size=0.2, random_state=42)

svr = SVR(kernel='rbf')
svr.fit(X_train, y_train)

y_svm_pred = svr.predict(X_test)

mse_svm = mean_squared_error(y_test, y_svm_pred)
print("Mean Squared Error:", mse_svm)

# Plotting y6 and y_svm_pred
plt.figure(figsize=(10, 6))
plt.plot(range(len(y_test)), y_test, label="Actual", marker='o', color='b')
plt.plot(range(len(y_test)), y_svm_pred, label="Predicted", marker='x', color='r')
plt.legend()
plt.title("Actual vs. Predicted Values")
plt.xlabel("Data Point")
plt.ylabel("Value")
plt.show()

```

```
Mean Squared Error: 0.02677290733636502
```

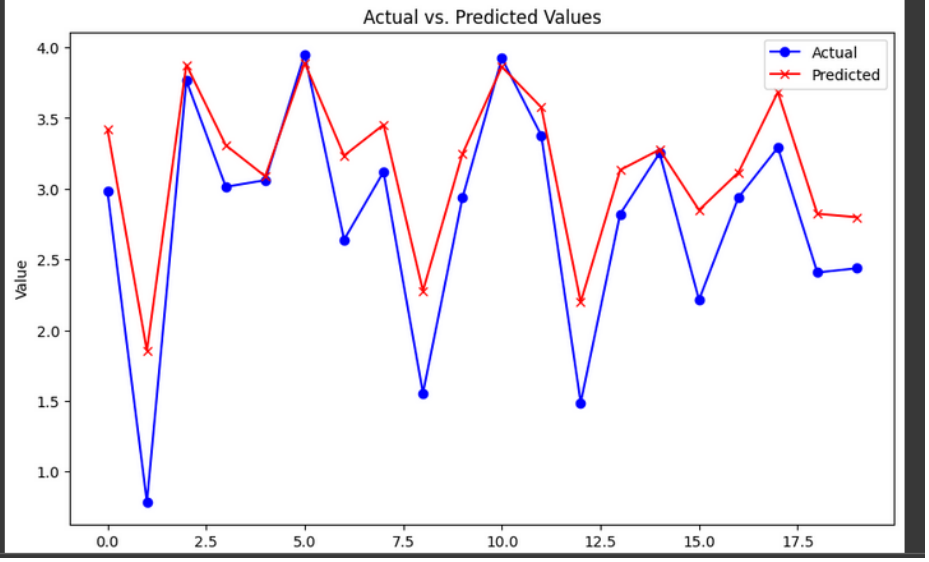


Using neural Netowork :

MSR was 0.20

```
nn = MLPRegressor(hidden_layer_sizes=(100, 80,50,25), max_iter=1000)
nn.fit(X_train, y_train)

y1_nn_pred = nn.predict(X_test)
mse_nn = mean_squared_error(y_test, y1_nn_pred)


print("Mean Squared Error:", mse_nn)

# Plotting y6 and y_svm_pred
plt.figure(figsize=(10, 6))
plt.plot(range(len(y_test)), y_test, label="Actual", marker='o', color='b')
plt.plot(range(len(y_test)), y1_nn_pred, label="Predicted", marker='x', color='r')
plt.legend()
plt.title("Actual vs. Predicted Values")
plt.xlabel("Data Point")
plt.ylabel("Value")
plt.show()
```

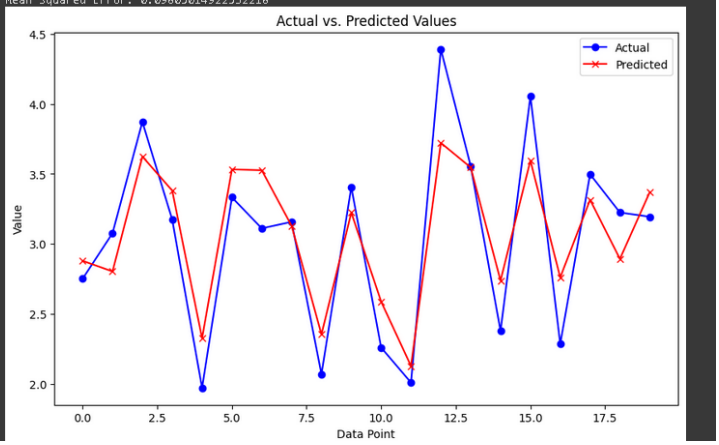
```
Mean Squared Error: 0.20302276965948715
```



Using KNN regressor: With 5 Neighbours . MSR was 0.09

```
X_train, X_test, y_train, y_test = train_test_split(X6, y6, test_size=0.2, random_state=42)

knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train)

y1_knn_pred = knn.predict(X_test)
mse_knn = mean_squared_error(y_test, y1_knn_pred)


print("Mean Squared Error:", mse_knn)
plt.figure(figsize=(10, 6))
plt.plot(range(len(y_test)), y_test, label="Actual", marker='o', color='b')
plt.plot(range(len(y_test)), y1_knn_pred, label="Predicted", marker='x', color='r')
plt.legend()
plt.title("Actual vs. Predicted Values")
plt.xlabel("Data Point")
plt.ylabel("Value")
plt.show()
```

```
Mean Squared Error: 0.09805014922532218
```



How To improve regression:

Generalization: We have to stop overfitting the data, so we need to regularize the models.

Identify outliers: Can Use RANSAC (sample from complete data build regression models remove outliers from it and then build a complete model).

Ensemble: We can combine two-three models and take average value on every prediction. For example, take two models SVR and Neural Network, take their average to predict.

Collab Link to codes:

1 Var: https://colab.research.google.com/drive/1xdstbi_yjYrss13Vzv4DHgCPN3wrIU53?usp=sharing

2 Var: https://colab.research.google.com/drive/1BJ6pRoasjRwKKtuebZEBGyvsUs5MGjpJ?usp=sharing

6 Var: https://colab.research.google.com/drive/1VDHtBwCZR5eIO_z7c7KYQCa2jc8oKqVb?usp=sharing