Bayesian Hyperparameter Optimization

Contributors
Abhishek, Guy, and Mark

Problem Definition

- Training Challenges with Multi-Layer Perceptron
- Grid Search Limitations
- What does Bayesian Optimization try to accomplish?

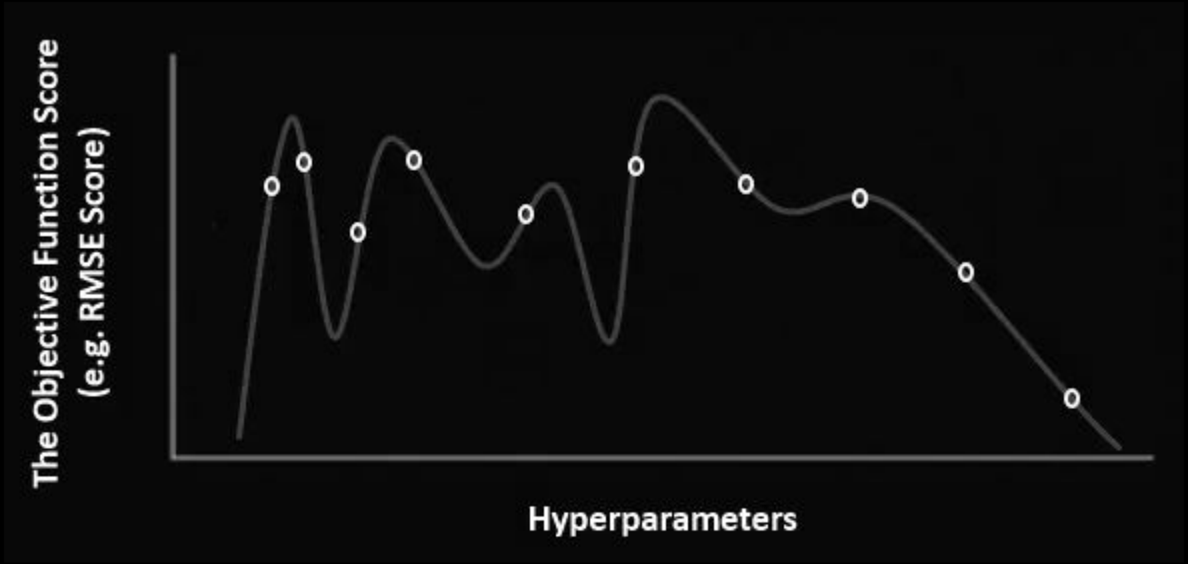
Gaussian Process Model

- Why are Gaussian Process (GP) models relevant?
- Foundation of Gaussian Process model:
  - Bayes Theorem

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

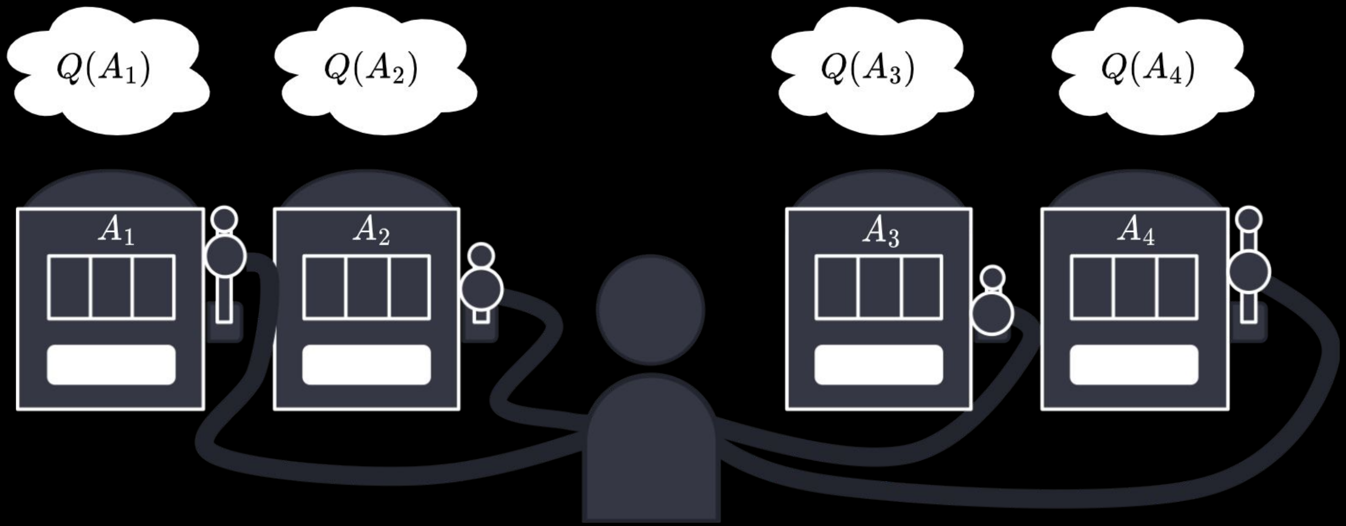
Bayesian Optimization

- What is Bayesian Optimization?
- How does it solve our problem?



Acquisition Function

- What is an Acquisition Function?
- Exploration vs. Exploitation



Probability of Improvement

- Focuses where the probability of improving the current best known value is highest
- Advantages:
  - Simple and easy to implement
  - Effective in noisy environments
- Limitations:
  - Tends to over-exploit known areas
  - May miss discovering better regions

$$PI(x) = \Phi\left(\frac{\mu(x) - f' - \xi}{\sigma(x)}\right)$$

Expected Improvement

- Selects points expected to most improve the current best value.
- Advantages:
  - Balances exploration and exploitation
  - Encourages selection of points with potential significant improvements
- Comparison:
  - More versatile than Probability of Improvement (PI) in finding optimal solutions
- Calculates probability of model

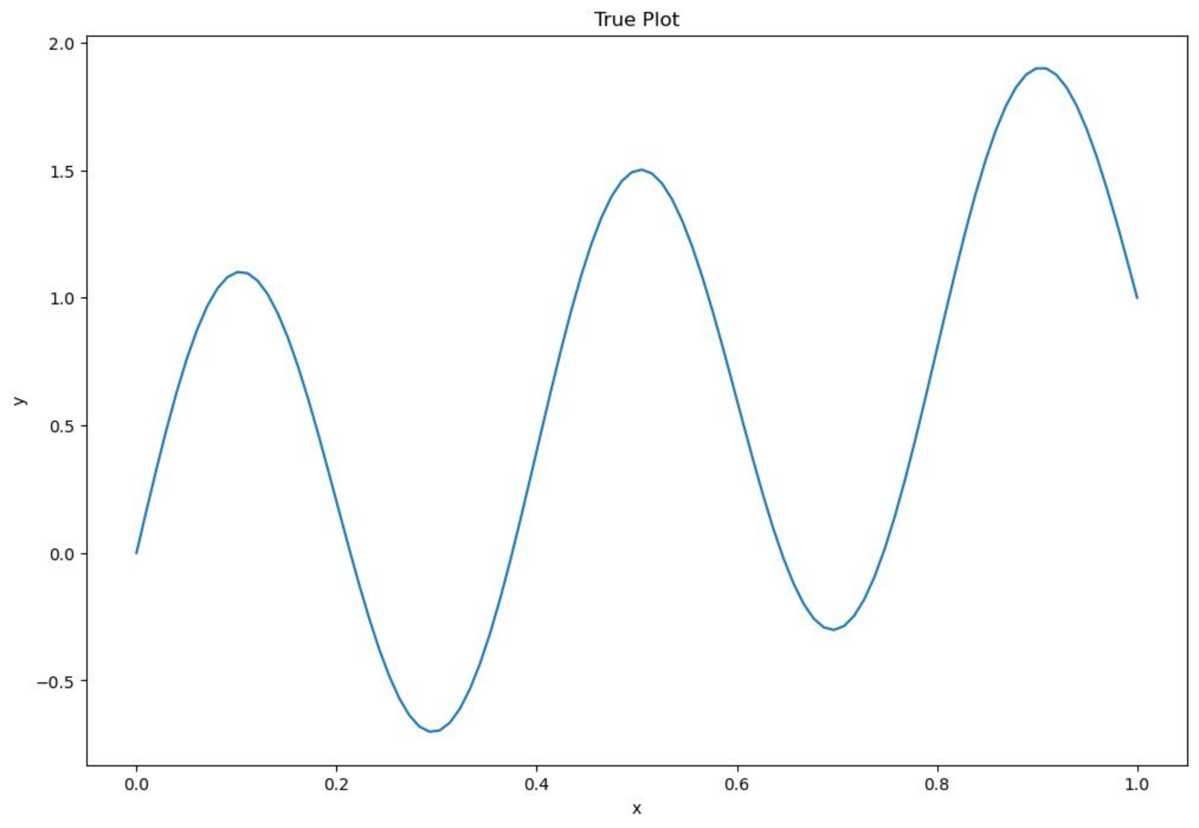
$$a_{EI}(x) = (\mu(x) - f' - \xi)\Phi\left(\frac{\mu(x) - f' - \xi}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{\mu(x) - f' - \xi}{\sigma(x)}\right)$$

Upper Confidence Bound

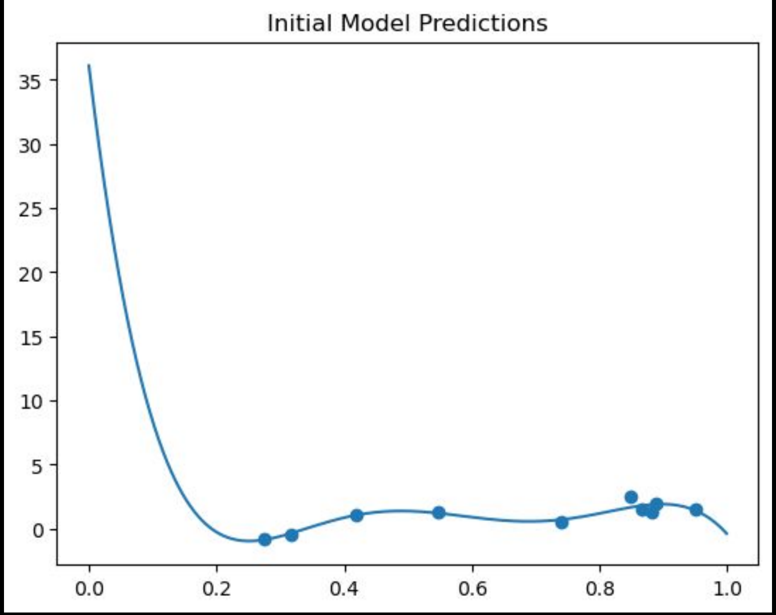
- What is the Upper Confidence Bound?
- Advantages:
  - Simpler to compute
  - Simple.
  - Robust function, mathematically guaranteed to converge to the global optimum.
- Disadvantages:
  - Performs poorly in high variance situations

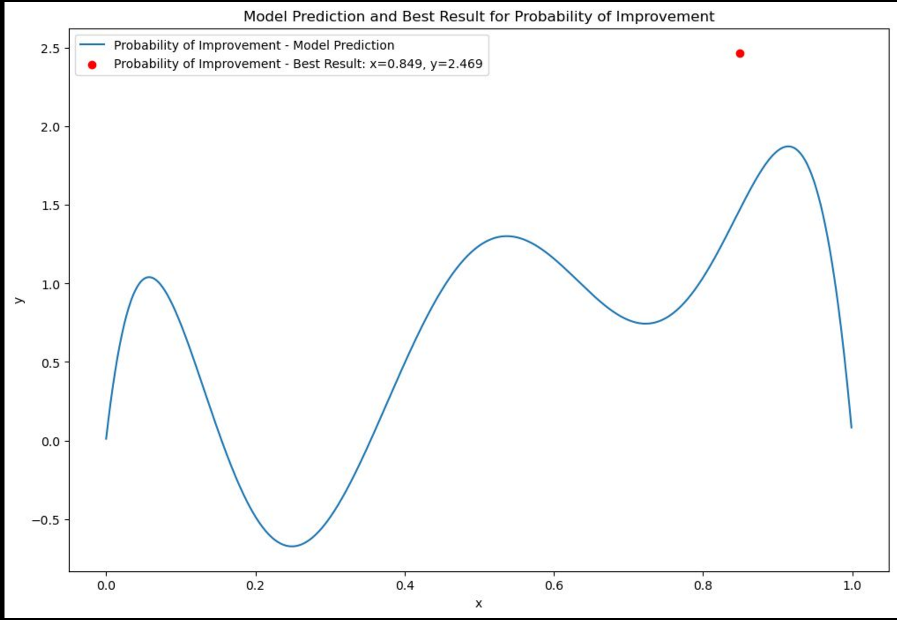
$$a(x;\ \lambda) = \mu(x) + \lambda\sigma(x)$$

Demo Results – True Plot

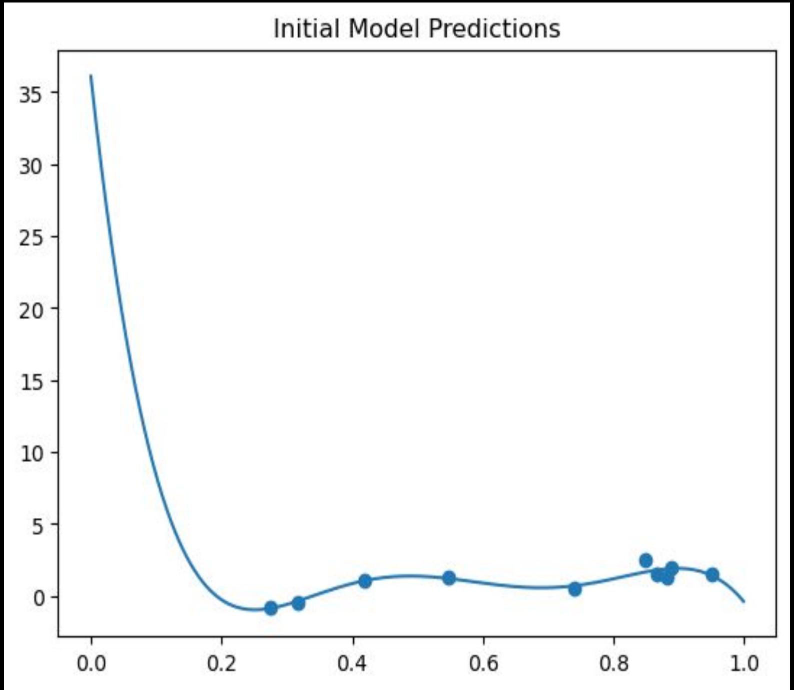


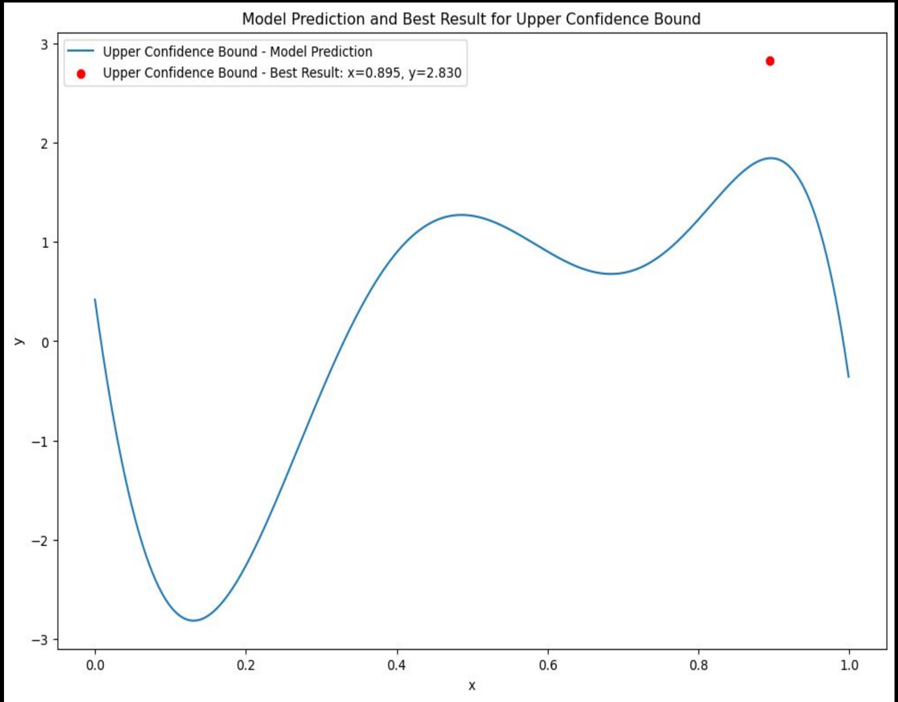
Demo Results – PI





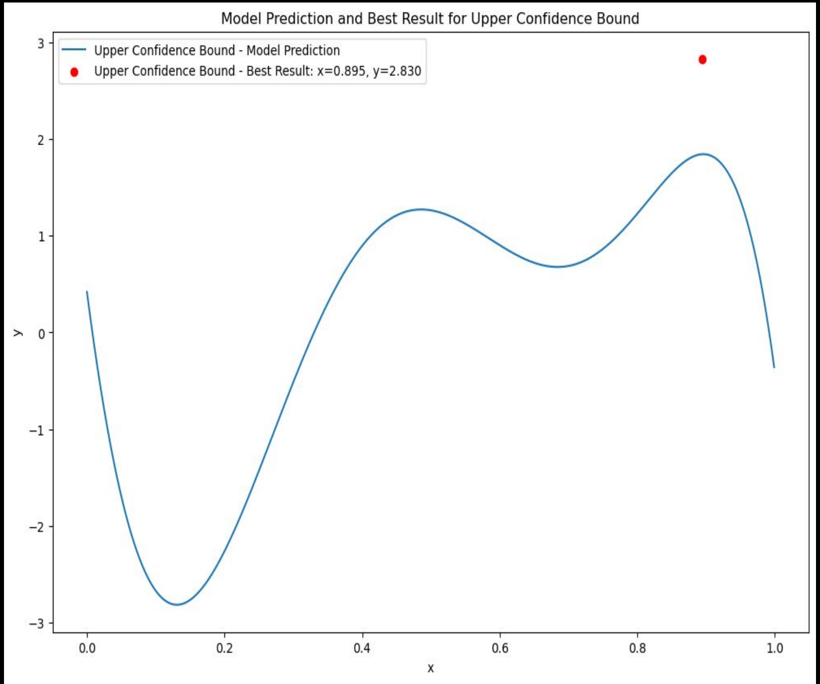
Demo Results – UCB

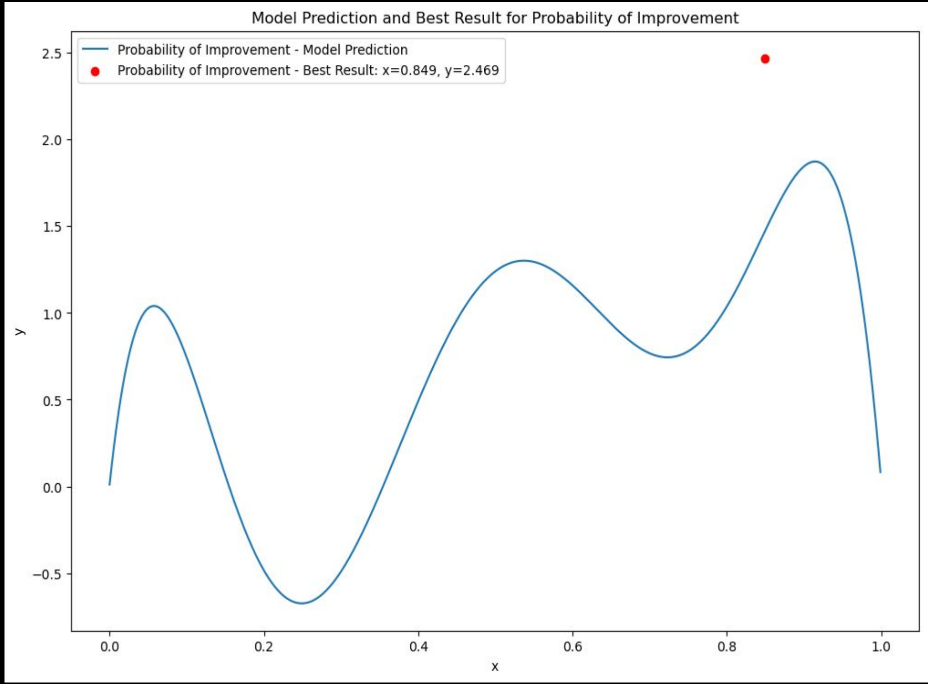




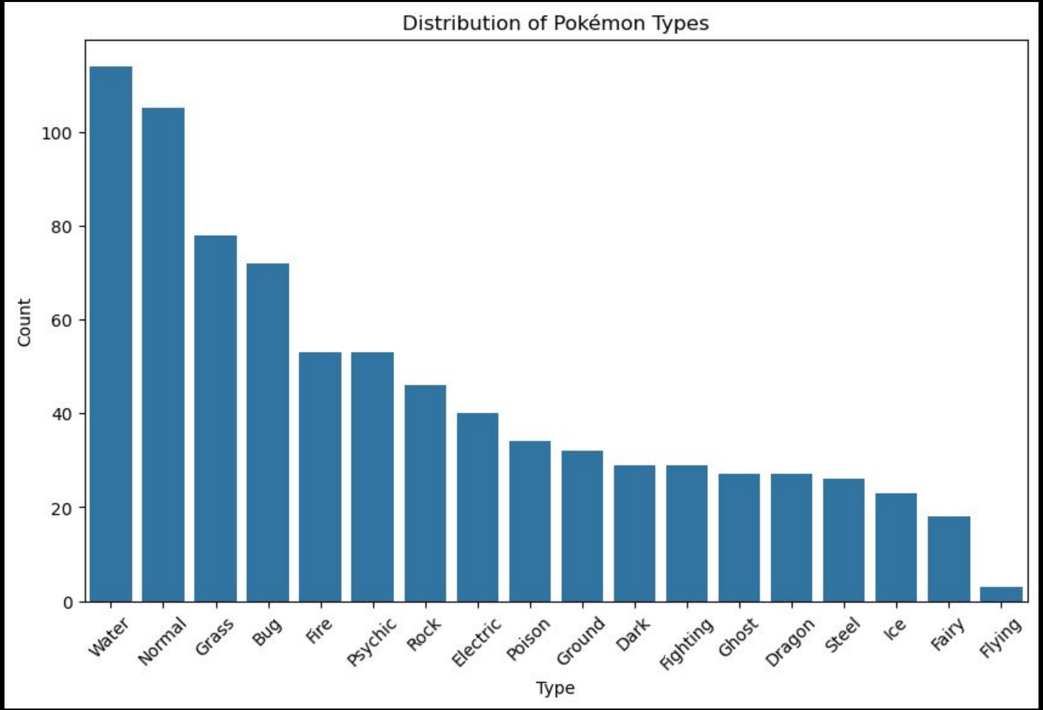
Demo Results

- UCB performed better than PI





Dataset Loading



Objective Function

```python
# Training function
def train_model_objective_func(hidden_layer_size, activation_function_ind, alpha_ind):
    activation_functions = ['identity', 'logistic', 'tanh', 'relu']
    alphas = np.logspace(0, 1, 100)
    activation_function = activation_functions[activation_function_ind]
    alpha = alphas[alpha_ind]
    model = MLPClassifier(hidden_layer_sizes=(hidden_layer_size, hidden_layer_size,),
                          activation=activation_function,
                          alpha=alpha)
    model.fit(X_train, y_train)
    return my_error_metric(model, X_train, y_train, X_test, y_test)[1], (hidden_layer_size, activation_function_ind, alpha_ind)
```

Sample Acquisition

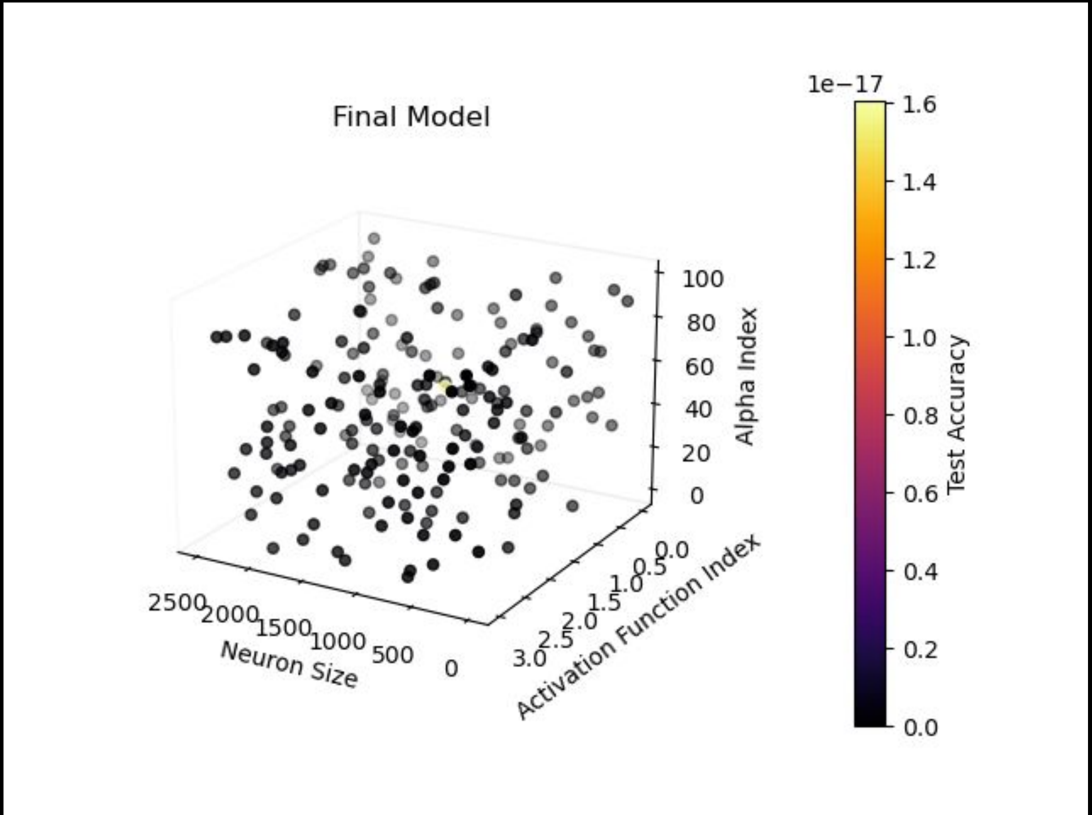
```python
def generate_sample():
    activation_functions = ['identity', 'logistic', 'tanh', 'relu']
    alphas = np.logspace(0, 1, 100)
    neuron_size = np.random.randint(1, 2501)
    acquisition_function = np.random.randint(0, len(activation_functions))
    alpha = np.random.randint(0, len(alphas))
    return neuron_size, acquisition_function, alpha


# define an acquisition function
def choose_acquisition(X, model):
    # Generate random samples for exploration
    samples = np.array([generate_sample() for _ in range(1000)])
    # Evaluate acquisition scores for each sample
    scores = upper_confidence_bound(X, samples, model)
    # Identify the sample with the highest score
    best_index = np.argmax(scores)
    return samples[best_index]
```

Hyperparameter Optimization

```python
for i in tqdm(range(50)):
    sample = choose_acquisition(X, pokemon_model_gp)
    np.append(X, sample)
    np.append(y, train_model_objective_func(*sample)[0])
    pokemon_model_gp.fit(X, y)
```

Final Result




```
Best Found Hyperparameters: 681 neurons, activation_function=relu, alpha=2.205130739903046
Test Accuracy: 0.3448275862068966
```