Project Update

**Research Progress:**

Particle swarm Optimization is an algorithm inspired by swarms of birds and fish in the wild. It was discovered by James Kennedy and Russel Eberhart in 1996. Originally the algorithm was written to imitate social behavior, but with some edits and simplification, it just turned into an optimization algorithm. PSO is what computer scientists call metaheuristic meaning it makes few to no assumptions on the problem it's given to optimize while it still going over large amounts of potential solutions.

In PSO the solutions are called particles, and the particles move around in a search space. The particles move around using a few formulas based on their velocity and position in reference to the particle’s best option from its own movements as well as global movements from other particles. The use of global and local references makes the particles move in a swarm-like movement toward the best solution. The process is repeated with the hope of finding an optional solution.

While the PSO may be seen as rudimentary compared to other algorithms, it serves a very unique purpose just like linear and logistic regression that play a role due to their simplicity and easy understandability. The PSO's ability to efficiently search a large, nonlinear space makes it well-suited for complex optimization problems where more advanced algorithms would be intractable or get stuck in local minima. Its stochastic nature also allows the PSO to explore the search space more thoroughly than gradient-based methods.

**Implementation Status:**

Good progress has been made in implementing the particle swarm optimization (PSO) algorithm in Python. The basic structure and logic of the algorithm have been coded, including initialization of the particle swarm, calculation of particle velocities and positions, and updating the global best solution. One challenge encountered was setting appropriate limits on particle velocity to prevent explosion. This was addressed by calculating a dynamic velocity clamp based on the current velocity and distance from the global best position. Another challenge was selecting good values for the acceleration coefficients c1 and c2 to balance exploration and exploitation. This was addressed through empirical testing of various coefficient values on benchmark functions. Overall, the core PSO algorithm is functioning as expected. Further work is needed to integrate adaptive inertia weight and velocity clamping methods to improve performance. Testing on some additional benchmark functions will help refine the parameters and confirm the algorithm is avoiding local optima efficiently before applying it to the complex real-world problem we intend to optimize. But good progress has been made toward a working PSO implementation.

**QA Testing:**

We iterated through many designs of our code and have resolved every code issue. We iterated through multiple designs of visualization to assess our models, which we struggled with a bit, but we figured out good ways to visualizer that with animations in matplotlib and read more papers on PSO to understand it. Moving forward if we encounter any issues we found its fairly easy to resolve them with research.

Psuedo Code:

