T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 498 BİTİRME ÇALIŞMASI

YAPAY ZEKA İLE EVRİMSEL SEÇİLİM SİMÜLASYONU

G201210007 – Furkan LİMAN

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ
Tez Danışmanı : Dr. Öğr. Üyesi Serap ÇAKAR KAMAN

2023-2024 Bahar Dönemi

T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

YAPAY ZEKA İLE EVRİMSEL SEÇİLİM SİMÜLASYONU

BSM 498 - BİTİRME ÇALIŞMASI

Furkan LİMAN

Fakülte Anabilim Dalı : BİLGİSAYAR MÜHENDİSLİĞİ

Bu tez .. / .. / … tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

…….	…….	…….
Jüri Başkanı	**Üye**	**Üye**

ÖNSÖZ

Hayatımıza olan etkisi sürekli artmakta olan yapay zekâ teknolojileri bilim dünyasında köklü değişimlere yol açmış, bilimin pek çok alanında problemleri çözmek için kullanılan güçlü bir araç haline gelmiştir. Biyolojik evrim ve seçilim baskılarının modellenmesi, geçmişe dönük tahmin veya bulguların simüle edilmesi verilerin büyüklüğü nedeniyle oldukça zor bir süreçti. Yapay zekanın büyük veriler üzerindeki başarılı ve hızlı tahlilleri, doğru yaklaşımlarla birlikte genetik süreçleri taklit ederek biyolojinin birçok problemine çözüm imkânı sağladığı görülmektedir.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

AF	: Aktivasyon Fonksiyonları
API	: Uygulama Programlama Arayüzü
DS	: Doğal Seçilim
YS	: Yapay Seçilim
YSA	: Yapay Sinir Ağları
YZ	: Yapay Zeka
RFR	: Random Forest Regressor
MQR	: MultiOutputRegressor
ete3	: Environment for Tree Exploration
PIL	: Python Imaging Library

ŞEKİLLER LİSTESİ

ÖZET

Anahtar kelimeler: Yapay Zeka, Evrimsel Seçilim, Simülasyon

Evrimsel seçilimin yüzlerce değişkene sahip doğa tarafından kaos ile yönetilmesi geçmiş çağlardaki canlılara dönük yapılan araştırmalardaki sonuçların eksik kalmasına sebep olmaktadır. Geçmiş çağlarda yaşanmış olabilecek veya gelecekte yaşanması mümkün olası çevrelerin canlılar üzerindeki etkilerinin simüle edilmesi hem geçmişin anlaşılmasına hem de gelecekte insanlığın karşılaşabileceği olası problemlerin öngörülmesini sağlayabilir. Yapay zekanın büyük veri gruplarındaki başarısı sayesinde bu çevrelerdeki uyum süreçlerini modellemek amacıyla bir simülasyon ortamı oluşturmayı hedeflenmektedir.

Bu çalışma, yapay zekanın doğal seçilim baskılarını incelemesi ve canlıların oluşturulan ortam içerisindeki davranışının tahmini üzerinedir. Oluşturulan ortamın ve canlıların etkileşimlerinin olası sonuçlarının hesabı için makine öğrenimi veya kural tabanlı yapay zekâ yöntemleri seçilebilir. Seçilen yöntem simüle edilecek bireylerin popülasyon içerisindeki uygunluğunu belirleyecek ve olası senaryoları çizecektir.

BÖLÜM 1. GİRİŞ

Doğal seçilim ve canlının çevre içerisindeki uyum sürecinin simülasyon üzerinde yapay zeka destekli bir şekilde hesaplanması hedeflenmektedir. Canlı bir ortamın sanal bir sistem üzerinde oluşturulması beraberinde sorunlar getirmektedir. Bireyin çevre ile olan etkileşiminin sınırları, oluşturulacak simülasyon üzerinde direkt bir etki göstermektedir. Bireyin sahip olacağı değişkenler ve metotlar canlı etkileşiminin temel karmaşasını ve seçilecek algoritmayı belirleyecektir.

Canlı çevrenin sıcaklık, nem, asiditesinden etkilenebilir. Sıcaklığın canlının direnç sınırlarından çıkması sonucunda canlı hayatını kaybeder. Canlı sıcağa uyum sağlayabilecek bir şekilde gelişmezse nesli tehlike altına girecektir [8]. Nem ve asidite de benzer etkilere sahiptir. Bu etkiler canlının gerçek dünya üzerinde karşılaşabileceği zorlukların temsiliyetlerinin bazılarıdır. Bu çevresel etkilerin kullanıcı tarafından kontrol edilebilmesi çeşitli senaryolar oluşturma imkânı vermektedir. Bu senaryolar gerçek hayatta yaşanan bir olayı simülasyon içerisinde test etmek amacı taşıyabilir ya da gerçekleştirilmek istenen bir deney senaryosu olabilir. Canlı topluluğunun senaryolar tarafından test edilmesi, gerçek hayatta yaşanabilecek sonuçlar hakkında öngörü kazanmaya yardımcı olacaktır.

Canlının çevre ile etkileşiminin yanı sıra diğer canlılar ile etkileşimi söz konusu olabilir. Bir canlının yemeğe ulaşmak yerine diğer canlılar ile yemek uğruna savaşabilir. Bu savaş canlılar arasındaki çeşitli özelliklere dayanacaktır.

Simülasyon tasarım itibariyle canlıların fenotip özelliklerinin etkileşimleri üzerinde durmaktadır. Genotipler araştırmanın ilerleyen safhasında dahil edilebilir.

Genotip özellikleri genomlar arasındaki bağlar belirler. Bu bağlar canlı için gerekli ürünlerin üretiminden sorumludur. Bir genom birden çok fenotip özelliği belirleyebildiği gibi bir fenotipi birden çok genom etkileyebilir. Buradaki bağlantıların sayısı sistem performansı ile ters orantılıdır. Genomlar arasındaki

ilişkinin uygulamaya dahili fenotip çıktılar üretebilir. Canlıların fenotip özelliklerinin çevreyle etkileşiminde genomlar önemli bir rol oynamaktadır.

Simülasyonlar gerçek ile yapay arasında bir örüntü oluşturmalıdır. Bir simülasyon ile gerçek arasındaki sınır, bağlama bağlı olarak değişebilir. Daha önce doğal seçilim mekaniğini inceleyen simülasyonlar da aynı sınırlar içerisine hapsolmuşlardır. Daha önce canlı hızı, görüş alanını gibi değerleri seçilim mekanikleri ile ele alan birkaç uygulama mevcuttur.

Justin Helps'in yaptığı Primer Learning çalışması bu alanda yapılmış sayılı projelerden biridir. Benzer bir simülasyon ortamı kurmayı hedeflese bile simülasyonun sınırları hız, yaş, görüş mesafesi, özelliklerinden dışarı çıkamamıştır. Canlılar benzer bir çevrede doğarlar, amaçları yiyeceğe ulaşmaktır. Popülasyon nesilden nesile incelenebilmektedir. Çevresel faktörler ve canlılar arasındaki etkileşimler göz ardı edilmiştir. Canlılar belirli bir açıyı görebilme yetisine sahip değil tüm çevrelerini görmektedirler. Canlıların popülasyon bilgilerine ve popülasyonun nicel kriterlerine dahil herhangi bir çıkarımda bulunmamış ve grafik sunmamıştır.

PhET Interactive Simulations, tarafından yapılan Natural Selection projesi, seçilim simülasyonunu eğitim açısından ele almıştır. Esas amacı çocuklara seçilim mekaniğini göstermek olan uygulamada canlılar iki renk çeşidine sahip tavşanlar ve kurtlar olarak belirlenmiş ve hiçbir spesifik özelliğe değinilmemiştir. Kurtlar ve tavşanlar arasındaki av avcı ilişkisi üzerinde durmuştur. Tüm Canlıların hızları ve özellikleri sabittir. Her canlı türü sadece bir animasyona sahiptir. Canlı popülasyonuna ait grafik sunar.

Diğer çalışmalardan farklı olarak bu çalışma sadece simülasyon ortamını değil aynı zamanda bu ortamdan alınan verilerle eğitilmiş bir yapay zeka uygulaması içermektedir. Bu yapay zeka uygulaması simülasyon sürecini hızlandırarak simülasyon ortamının sınırlarına çıkan olaylar ve verilerle çalışma imkanı sağlamaktadır. Ayrıca simülasyon sürecindeki yaşanabilecek performans sorunlarının da önüne geçmeyi amaçlamıştır.

BÖLÜM 2. SİSTEMATİK YAKLAŞIM

2.1. Doğal Seçilimin Sanallaştırılması

Doğa içerisinde bir canlıya uygulanan seçilimin sınırları, çevresi ve genleri tarafından çizilir. Yaşadığı çevredeki sıcaklık, nem, hava durumu, topografya, avcı türleri, besine ulaşabilme yöntemleri, canlının fenotip ve genotipi gibi birçok değişken bireyin yaşamını etkilemektedir [1]. Canlının bulundurduğu genotip ve fenotipler canlının seçilim baskılarını ve popülasyondaki çeşitliliği belirler [2]. Her canlı için her an değişen bu özellikler, doğadaki çeşitliliğin fazlalığı sebebiyle canlının doğrudan simülasyona eklenmesi önünde engel teşkil eder.

Canlının simülasyona eklenmesi sürecinde canlıya atfedilen değişkenler ve özellikler simülasyonun tutarlılığı için basit seçilmelidir. Oluşturulacak ortamın tek hücreli veya az cinsli canlılardan oluşması belirli bir karakteristik özelliğin bireye ne kadar etkisi olduğunu göstermek ve simülasyonun tutarlılığı açısından önemlidir [3].

2.2. Yapay Zeka Entegrasyonu

Simülasyonun çalıştırılması ortamın canlı takibi ve izlenmesi açısından gereklidir ancak simülasyon yapısı gereği daha kapsamlı deneyler yapılmak istendiğinde daha büyük veri setleri ve daha uzun süren çalışma zamanlarına ihtiyaç duyabilir. Bu bazı sistemlerde performans sorunlarına sebep olabilir. Ayrıca uzun süren deney senaryolarının simülasyon ortamında takibi de uzun sürmektedir. Yapay zeka bu ve benzeri problemlere çözüm olarak daha efektif ve hızlı bir şekilde deney sonuçlarına ulaşmamızı sağlar. Çeşitli senaryolarla test edilmiş simülasyon ortamının çıkarttığı verilerle eğitilmiş olan yapay zeka, simülasyonu çalıştırmaya gerek olmadan deney sonucunu tahmin etmektedir. Bu yöntem sistemin performans ve zaman tasarrufu açısından oldukça önemlidir. Bu süreçte kullanılacak model, Random Forest Regressor (RFR) ve MultiOutputRegressor kombinasyonudur.

2.2.1. Random Forest Regressor (RFR)

Random Forest, temel olarak bir ensemble (topluluk) yöntemidir ve birçok karar ağacının birleşiminden oluşur. Bu yöntemde, her bir karar ağacı veri setinin farklı bir alt kümesi üzerinde eğitilir ve sonrasında tüm ağaçların tahminlerinin ortalaması alınarak nihai sonuç elde edilir. Bu yaklaşım, bireysel karar ağaçlarının aşırı uyum (overfitting) yapma riskini azaltır ve genel olarak daha yüksek doğruluk sağlar.
Bu projede ise, belirli bir gün ve mutasyon oranı verildiğinde simülasyon ortamındaki diğer değişkenleri tahmin etmek amacıyla Random Forest Regressor kullanılmıştır. Bu model, veri setimizdeki çoklu hedef değişkenler için tahmin yapabilme kabiliyeti nedeniyle tercih edilmiştir.

Bu projede kullanılan Random Forest Regressor modeli, Scikit-learn kütüphanesinin MultiOutputRegressor sınıfı ile birleştirilmiştir. Bu sayede, birden çok hedef değişken için tahmin yapabilen bir model oluşturulmuştur. Modelin performansını değerlendirmek için Mean Squared Error (MSE) metriği kullanılmıştır. MSE, tahmin edilen değerler ile gerçek değerler arasındaki farkların karesinin ortalamasını alarak modelin doğruluğunu ölçer.

Random Forest Regressor, tahmin yapmak için kullanılan bir regresyon modelidir ve şu adımları takip eder:

1. Veri Setinin Rastgele Alt Kümelere Ayrılması: Model, eğitim veri setini rastgele seçilen alt kümelere böler. Bu alt kümeler bootstrap örneklem olarak adlandırılır ve her bir ağaç bu alt kümeler üzerinde eğitilir.
2. Karar Ağaçlarının Eğitimi: Her bir alt küme üzerinde ayrı ayrı karar ağaçları eğitilir. Bu süreçte, her bir ağaç belirli bir dizi karar kuralları kullanarak tahmin yapmayı öğrenir.
3. Tahminlerin Birleştirilmesi: Tüm karar ağaçları eğitildikten sonra, yeni bir veri noktası için tahmin yapılmak istendiğinde, her bir ağaç bağımsız olarak tahmin yapar. Bu tahminler birleştirilir ve ortalaması alınarak nihai tahmin elde edilir.

Random Forest Regressor'ın Avantajları:

- Genel Performans: Birden fazla karar ağacının birleşiminden oluştuğu için genellikle tek bir karar ağacından daha yüksek doğruluk sağlar.
- Aşırı Uyum Riskinin Azaltılması: Farklı alt kümeler ve rastgele özellik seçimi sayesinde modelin aşırı uyum yapma riski azalır.
- Esneklik: Hem sınıflandırma hem de regresyon problemlerinde etkili bir şekilde kullanılabilir.
- Öznitelik Öneminin Belirlenmesi: Model, hangi özelliklerin tahminler üzerinde daha etkili olduğunu belirleyebilir ve bu bilgiler modelin yorumlanabilirliğini artırır.

2.2.2. MultiOutputRegressor

MultiOutputRegressor, scikit-learn kütüphanesinde bulunan ve çoklu hedef değişkenler (multi-output) için tahmin yapabilen bir regresyon modelidir. Bu model, her bir hedef değişken için ayrı ayrı regresyon modelleri eğitir ve tahmin yapar. Böylece, birden çok bağımlı değişkenin aynı anda tahmin edilmesi mümkün hale gelir.

MultiOutputRegressor, her hedef değişken için ayrı bir regresyon modeli oluşturur ve bu modelleri bağımsız olarak eğitir. Tahmin aşamasında, her model kendi hedef değişkeni için tahminlerde bulunur ve sonuçlar birleştirilir. Bu, her bir hedef değişkenin en iyi şekilde tahmin edilmesini sağlar.

MultiOutputRegressor kullanmanın çalışmaya getirdiği avantajlar:

- Çoklu Hedef Değişkenler: Birden fazla hedef değişkeni aynı anda tahmin edebilir.
- Esneklik: Her bir hedef değişken için farklı regresyon modelleri kullanılabilir.
- Kolay Entegrasyon: Scikit-learn kütüphanesi ile kolayca entegre edilebilir ve kullanımı oldukça basittir.

MultiOutputRegressor kullanmanın çalışmaya getirdiği dezavantajlar:

- Hesaplama Maliyeti: Her bir hedef değişken için ayrı model eğitildiği için, hesaplama maliyeti ve eğitim süresi artabilir.
- Model Karmaşıklığı: Çok sayıda hedef değişken için model oluşturulması, modeli daha karmaşık hale getirebilir.

2.3. Python

Python, geniş bir kullanıcı kitlesine hitap eden, yüksek seviyeli bir programlama dilidir. İsmini Monty Python adlı bir İngiliz komedi grubundan alan Python, basit ve okunabilir sözdizimi ile öne çıkar. Bu dil, özellikle öğrenmesi kolay olması ve geniş bir kütüphane ekosistemine sahip olması nedeniyle öğrenciler, hobi programcıları ve endüstri profesyonelleri arasında popülerdir.

Python'un belirgin özelliklerinden biri, dilin sadeliği ve esnekliğidir. Okunabilirlik ilkesine dayanan bir tasarım felsefesi olan "Pythonic" yaklaşım, kodun anlaşılabilir olmasını teşvik eder. Bu, programcılara kodlarını daha hızlı yazma ve bakımını yapma imkanı tanır.

Python, geniş bir standart kütüphane içerir ve aynı zamanda üçüncü taraf kütüphanelerin geniş bir ekosistemine sahiptir. Bu, çeşitli uygulama alanlarında kullanılabilen zengin bir araç seti sağlar. Veri analizi, yapay zeka, web geliştirme, bilimsel hesaplamalar ve daha birçok alanda Python yaygın bir şekilde kullanılmaktadır.

Ayrıca, Python'un açık kaynak olması ve geniş bir topluluğa sahip olması, sürekli gelişen bir dil olmasını sağlar. Bu, Python kullanıcılarının sürekli güncel kalmalarını ve en yeni teknolojilere kolayca erişebilmelerini sağlar.

Sonuç olarak, Python, basitliği, okunabilirliği ve geniş bir ekosistemle desteklenen esnekliği sayesinde birçok programcının tercih ettiği ve öğrenmeye başladığı bir programlama dilidir.

2.4. VPython

VPython, Python programlama dilinde üç boyutlu grafikler ve animasyonlar oluşturmak için kullanılan güçlü bir kütüphanedir. Bu kütüphane, bilimsel ve fiziksel konularda karmaşık kavramları daha iyi anlamak için etkileşimli ve görsel bir araç sağlar. VPython, öğrencilere ve araştırmacılara soyut matematiksel veya fiziksel kavramları somut ve görsel bir şekilde deneyimleme imkânı sunar. Örneğin, bir üç boyutlu sahne oluşturarak nesnelerin konumlarını, şekillerini ve renklerini belirlemek oldukça basittir. Ayrıca, bu nesneleri hareket ettirme, döndürme ve animasyonlar oluşturma gibi işlemleri gerçekleştirmek de kullanıcı dostu bir şekilde sağlanır. Bu özellikler, öğrencilerin ve araştırmacıların soyut kavramları daha iyi kavramalarına yardımcı olurken, aynı zamanda bilimsel projelerde ve sunumlarda etkileyici görsel içerikler oluşturmak için ideal bir araçtır.

2.5. Numpy

NumPy, Python dilinde sayısal hesaplamaları kolaylaştıran temel bir kütüphanedir. Bu kütüphane, çok boyutlu dizilerle çalışarak veri manipülasyonunu ve matematiksel işlemleri optimize eder. Özellikle büyük veri setleri ve karmaşık matris işlemleri üzerinde etkili bir performans sergiler.

Sayısal uygulamalarda yaygın olarak kullanılan NumPy, veri bilimi, yapay zeka ve mühendislik alanlarında önemli bir rol oynar. Çok boyutlu dizilerle çalışmak, veriyi daha düzenli ve anlaması kolay hale getirir. Ayrıca, matematiksel işlemleri hızlı ve verimli bir şekilde gerçekleştirmek için tasarlanmış olması, analiz ve modelleme süreçlerini kolaylaştırır.

Rastgele sayı üretimi ve "broadcasting" gibi özellikler, çeşitli uygulamalarda kullanım esnekliği sağlar. NumPy, alt seviyede yapılan optimizasyonlar sayesinde yüksek performans sunar, bu da büyük veri setleri üzerinde hızlı hesaplamalar yapma avantajı sağlar.

2.6. Pandas

Pandas, Python programlama dilinde veri analizi ve veri manipülasyonu için kullanılan güçlü bir kütüphanedir. İsmine "panel data" ve "data frame" terimlerinden türetilmiş olup, özellikle yapılandırılmış veri üzerinde çalışmak için geliştirilmiştir. Pandas, çeşitli veri yapıları ve veri analiz araçları sunarak, veri bilimcilerin ve analistlerin veriyle daha etkili ve verimli bir şekilde çalışmasına olanak tanır.

Bu çalışmada Pandas kütüphanesi, yapay zeka modelinin eğitimi ve uygulanması sırasında veri tabloları oluşturmada ve verilerin düzenlenmesinde kullanıldı.
Çalışmaya getirdiği avantajlar:

- Hız ve Verimlilik: Büyük veri setleriyle çalışırken hızlı ve verimli performans sunar.
- Esneklik: Farklı veri formatlarıyla çalışabilir ve çeşitli veri manipülasyon işlemleri yapabilir.
- Kolay Kullanım: Python ile uyumlu ve kolay öğrenilebilir bir sözdizimine sahiptir.
- Geniş Topluluk ve Destek: Aktif bir kullanıcı topluluğu ve kapsamlı dokümantasyonu ile güçlü destek sağlar.

2.7. Environment for Tree Exploration (ete3)

Environment for Tree Exploration (ete3) biyolojik ağaçların oluşturulması, manipülasyonu ve analizi için kullanılan bir Python kütüphanesidir. ete3 özellikle filogenetik ve taksonomik ağaçların işlenmesi için tasarlanmıştır, ancak genel olarak herhangi bir hiyerarşik veri yapısıyla çalışabilir. Bu çalışmada canlıların genetik değişimlerini, mutasyonlarını ve evrimlerini takip etmek amacıyla kullanılmıştır. Yeni oluşmuş türlerin hangi genden geldiğinin takibi ve popülasyondaki yoğunluklarının takibini yapmaktadır. Bu çalışmada da yararlandığımız bazı özellikleri:

- Ağaç Görselleştirme: ete3, filogenetik ağaçları ve diğer hiyerarşik verileri görselleştirmek için güçlü araçlar sunar. Bu görselleştirmeler kullanıcıların ağaç yapılarını daha iyi anlamalarına yardımcı olur.

- Ağaç Manipülasyonu: Kütüphane, ağaçların kolayca oluşturulması, değiştirilmesi ve analiz edilmesi için çeşitli araçlar sağlar. Ağaç düğümleri ve dalları üzerinde karmaşık işlemler gerçekleştirilebilir.
- Filogenetik ve Taksonomik Analizler: ete3, filogenetik ağaçların oluşturulması ve analizi için birçok araç sunar. Bu, biyolojik araştırmalar için oldukça faydalıdır.
- Veri Entegrasyonu: ete3, NCBI taxonomy, Newick ve PhyloXML gibi yaygın biyoinformatik veri formatlarını destekler.

Bu çalışmadaki kullanım Alanları

- Filogenetik Ağaçların Oluşturulması ve Analizi: Biyolojik türlerin evrimsel ilişkilerini incelemek için kullanılır.
- Taksonomik Araştırmalar: Türlerin sınıflandırılması ve adlandırılması çalışmalarında kullanılır.
- Veri Görselleştirme: Hiyerarşik veri yapılarının görselleştirilmesi için genel amaçlı bir araç olarak kullanılabilir.

2.8. Plotly

Plotly, Python, R ve diğer programlama dilleri için açık kaynaklı grafik kütüphaneleri sunan bir platformdur. Plotly'nin Python versiyonu olan plotly.py, veri görselleştirme için güçlü ve esnek araçlar sağlar. Bu kütüphane, statik, dinamik ve interaktif grafikler oluşturmanıza olanak tanır.

Bu çalışmada plotly kütüphanesi, deney sonucunda canlılar üzerindeki değişimin radar grafik formatında görselleştirmesi amacıyla kullanılmıştır.

PIL kütüphanesinin çalışmaya getirdiği kolaylıklar ve özellikleri:

- Kolay Kullanım: Plotly, kullanımı kolay bir API sunar ve birkaç satır kodla karmaşık grafikler oluşturmanıza olanak tanır.
- Interaktif Grafikler: Grafikler üzerinde yakınlaştırma, kaydırma, veri noktalarına tıklama gibi etkileşimli özellikler sağlar.

- Çeşitli Grafik Türleri: Çizgi grafikleri, bar grafikleri, pie grafikleri, scatter plotlar, 3D grafikler ve daha fazlasını destekler.
- Çevrimiçi ve Çevrimdışı Kullanım: Grafiklerinizi çevrimiçi olarak Plotly sunucusunda saklayabilir veya yerel olarak HTML dosyaları olarak kaydedebilirsiniz.
- Entegrasyonlar: Plotly, Jupyter Notebook, Dash ve diğer veri analiz ve bilimsel hesaplama kütüphaneleri ile entegre çalışır.

2.9. Matplotlib

Matplotlib, Python programlama dilinde grafikler ve görselleştirmeler oluşturmak için kullanılan popüler bir kütüphanedir. John Hunter tarafından geliştirilen bu kütüphane, veri analizi ve bilimsel hesaplamalarla uğraşan araştırmacılar ve mühendisler için son derece kullanışlıdır. Matplotlib'in temel amacı, kolay ve esnek bir şekilde yüksek kaliteli grafikler oluşturmaktır.

Bu çalışmada Matplotlib kütüphanesi, popülasyon verilerinin takibi ve canlıların özelliklerinin günden güne grafiklendirilmesi için kullanılmıştır.

Matplotlib, birçok farklı grafik türünü destekler, bunlar arasında çizgi grafikleri, bar grafikleri, histogramlar, pasta grafikleri, dağılım grafikleri ve daha fazlası bulunur. Kütüphane, kullanıcıların veri görselleştirmelerini kişiselleştirmelerine olanak tanır. Grafiklerin başlıkları, eksen etiketleri, renkleri, çizgi stilleri ve diğer birçok özellik kolayca ayarlanabilir.

Matplotlib kütüphanesinin çalışmaya getirdiği kolaylıklar ve özellikleri:

- Veri Analizi ve Görselleştirme: Matplotlib, veri bilimcileri ve analistler tarafından verilerin görselleştirilmesi ve analiz edilmesi için yaygın olarak kullanılır. Özellikle büyük veri setlerinin anlaşılması ve sunulmasında önemli bir rol oynar.
- Bilimsel Hesaplamalar: Matplotlib, bilimsel hesaplamalar yapan araştırmacılar tarafından deney sonuçlarını ve matematiksel modelleri görselleştirmek için kullanılır.

- Makine Öğrenimi: Makine öğrenimi modellerinin performansını değerlendirmek ve sonuçlarını görselleştirmek için de Matplotlib sıkça tercih edilir.

2.10. Math

Python programlama dili, bilimsel hesaplamalar ve mühendislik uygulamaları gibi matematiksel işlemleri gerçekleştirmek için geniş bir standart kütüphane sunar. Bu kütüphanelerden biri olan math kütüphanesi, çeşitli matematiksel fonksiyonlar ve sabitler sağlar. Bu metin, Python math kütüphanesinin özelliklerini ve kullanımını açıklamaktadır.

Bu çalışmada Math kütüphanesi, canlılar arasındaki mesafeler ve canlılar üzerindeki vektörel işlemlerin hesaplanması sırasında kullanıldı.

2.11. Json

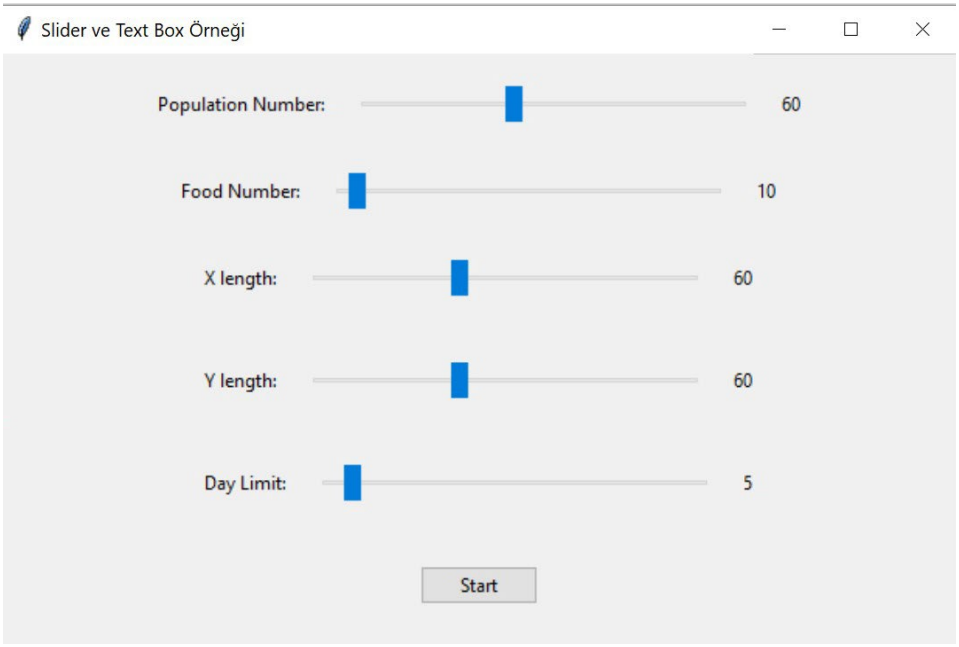
JSON (JavaScript Object Notation), veri değişimi için kullanılan hafif ve kolay anlaşılabilir bir formattır. JSON, özellikle web uygulamalarında veri iletimi için yaygın olarak kullanılmaktadır. JSON formatı, insan tarafından okunabilir ve yazılabilir olmasının yanı sıra, makine tarafından da kolayca ayrıştırılabilir ve oluşturulabilir. Bu kütüphane, JSON verilerini ayrıştırma (parse) ve JSON formatında veri oluşturma (serialize) işlemlerini kolaylaştırır. Çalışmada JSON kütüphanesi deney sonucundaki verilerin yapay zeka modelinde kullanılmak üzere saklanması amacıyla kullanılmaktadır.

2.12. Python Imaging Library (PIL)

Python Imaging Library (PIL), Python programlama dili için güçlü bir görüntü işleme kütüphanesidir. İlk olarak 1995 yılında Fredrik Lundh tarafından geliştirilen PIL, çeşitli görüntü formatlarını açma, işleme ve kaydetme yetenekleri sunar. Bu kütüphane, görüntüleri düzenlemek, filtreler uygulamak ve çeşitli analizler yapmak için kapsamlı araçlar sağlar.

BÖLÜM 3. UYGULAMA

3.1. Canlıların ve Ortamın Kurulması

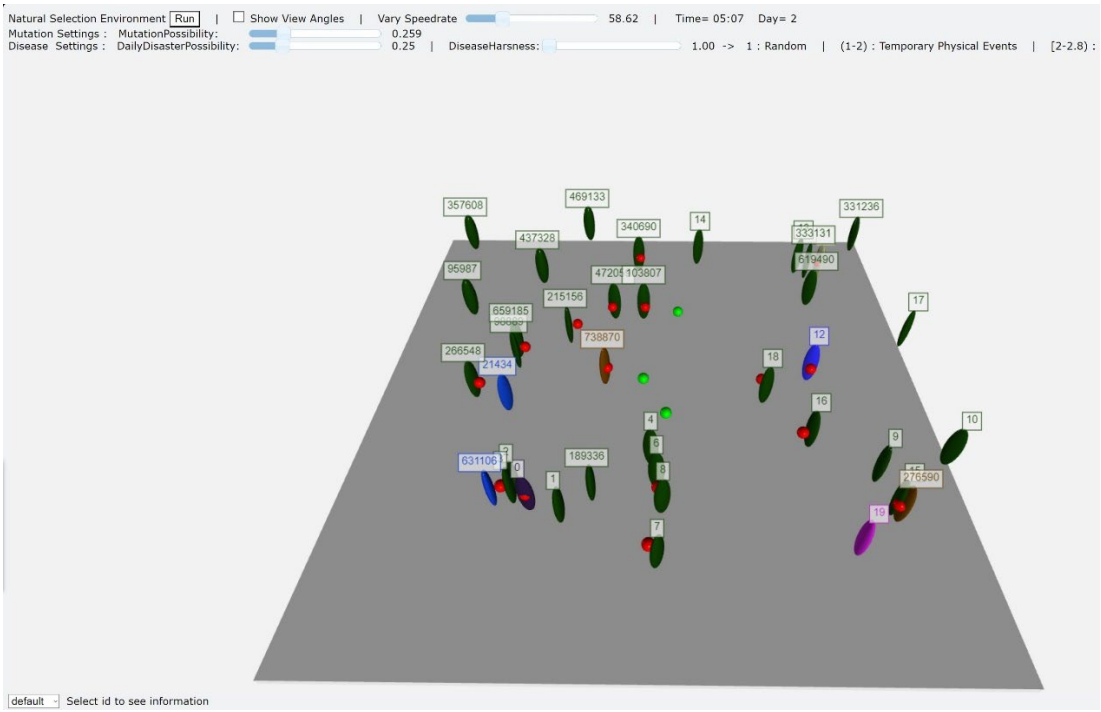


Şekil 3.1. Başlangıç penceresi

Uygulama çalıştırıldığında kullanıcıyı başlangıç penceresi karşılar (Şekil 3.1.).

Bu pencere:

- Başlangıçta bulunacak canlı sayısı
- Günlük ortama dağıtılacak yemek sayısını
- Simülasyon ortamının fiziksel boyutlarını
- Deneyin kaç süreceğini ayarlamaya olanak sağlar.



Şekil 3.2. Arayüz ve simülasyonun çalışması

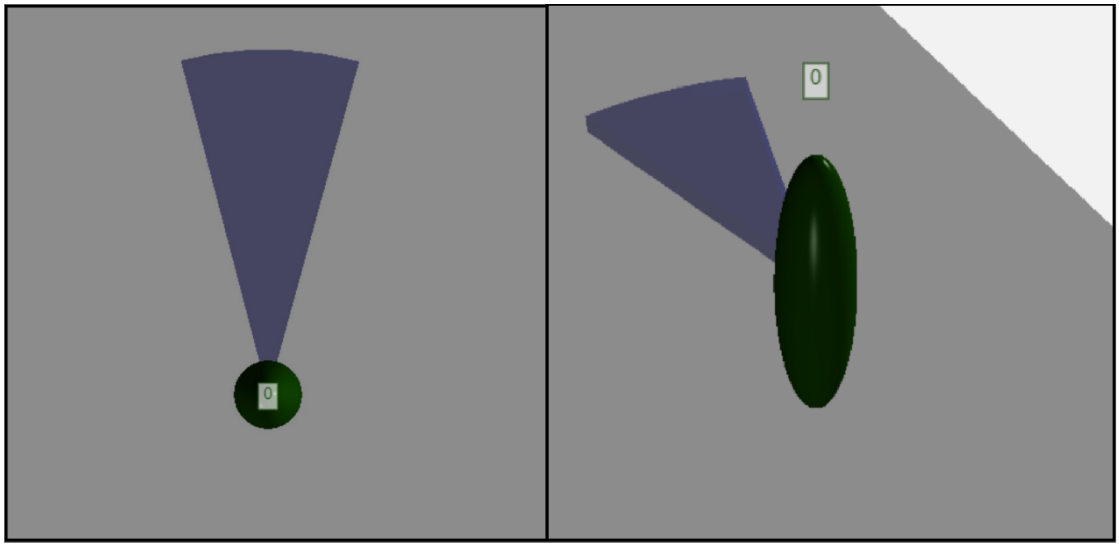
İstenilen başlangıç değerlerine göre ortamın ve canlıların gösterimi için VPython kütüphanesi yardımı ile simülasyon ortamı görsel olarak oluşturulur (Şekil 3.2.).

Başlayan simülasyon ortamı Şekil 3.2.'deki gibidir. Arayüzde kullanıcıya sunulan seçenekler:

- Show View Angles: Canlıların görüş açılarını gösterir.
- Vary Speedrate: Simülasyon hızını değiştirir.
- Mutation Settings: Gün sonunda canlılara uygulanacak mutasyon olasılığını belirler.
- Daily Disaster Possibility: Bir sonraki gün ortamda bir felaket gerçekleşme olasılığını belirler.
- Disease Harssness: Gerçekleşecek olayın türünü ve şiddetini belirler. 1 ve 3 arasında değer verilebilir.
    - 1 : rastgele bir olay belirler.
    - (1-2) : Geçici etkilere sahip olaylar oluşturur.
    - [2-2.8) : Viral hastalıklar oluşrurur.
    - [2.8-3] : Ölümcül hastalıklar oluşturur.

Canlılar gün döngüleri ile yaşamaktadır her günün amacı gün bitmeden yemeğe ulaşabilmektir. Gün sonunda yemeğe ulaşamayan canlılar silinir, yemeğe ulaşan canlılar ise çoğalırlar. Gün sonunda çoğalan canlılar genetik özelliklerini sonraki gün döngülerine aktarmış olur. Gün döngüleri sırasında kullanıcının isteği doğrultusunda çeşitli olaylar gerçekleşebilir.

3.1.1. Canlılar



Şekil 3.3. Canlının üstten ve yandan görüntüsü

Canlılar ‘Creature’ Class’ı içerisinde tanımlandı. Her canlı genetik ve fiziksel değerlere sahipler.

Genetik değerler canlının simülasyon ortamındaki etkileşimleri etkileyerek deney süresince canlıya avantaj veya dezavantaj sağlayabilecek şekilde tasarlandı.

Fiziksel değerler genellikle genetik değerlerin simülasyon ortamına yansımasıdır ve kullanıcının deneyimi için tasarlanmıştır. Seçilim baskısına bir etkisi yoktur.

Canlıların sahip olduğu genetik değerler:

- Speed (hız)
- Vision (görüş açısı)
- VisionRadius (görüş mesafesi)
- Immunity (bağışıklık)
- Durability (dayanıklılık)

Canlıların sahip olduğu fiziksel değerler:

- Id
- Konum
- Boyut
- Renk
- Açlık

Bu genetik değerlerin birleştirilmesiyle canlıya bir gen atfedilir. Gen mutasyona uğrayabilen ve gelecek nesillere aktarılabilen özelliktir. Genetik özellikler canlının simülasyon ortamındaki etkileşimlerini değiştirir. Genetik özelliklerin alabileceği minimum ve maksimum değerler kullanıcı tarafından belirlenir. Bu sayede canlı hiçbir genetik işlemin sonucunda fiziksel sınırların dışına çıkamaz.

Genetik özellikler ve sınırları:

- Speed (hız): Canlının simülasyon ortamındaki hızıdır. Minimum değeri varsayılan olarak 0.2 birim, maksimum değeri 2 birim olarak belirlenmiştir.
- Vision (görüş açısı): Canlının simülasyon ortamında görebileceği açı değeridir. Minimum değeri varsayılan olarak 10°, maksimum değeri 90° olarak belirlenmiştir.
- VisionRadius (görüş mesafesi): Canlının simülasyon ortamında görebileceği uzaklık değeridir. Minimum değeri varsayılan olarak 10 birim, maksimum değeri 30 birim olarak belirlenmiştir.
- Immunity (bağışıklık): Canlının sahip olduğu bağışıklık proteinlerini temsil eder. Gün içerisinde kullanıcı isteğine bağlı olarak çeşitli viral hastalıklar gerçekleşebilir. Canlı viral hastalığa anti bağışıklık proteini barındırmıyorsa

hastalığa kapılır. Hastalık durumu canlının kondisyon ve yapısına göre çeşitli etkiler bırakabilir.

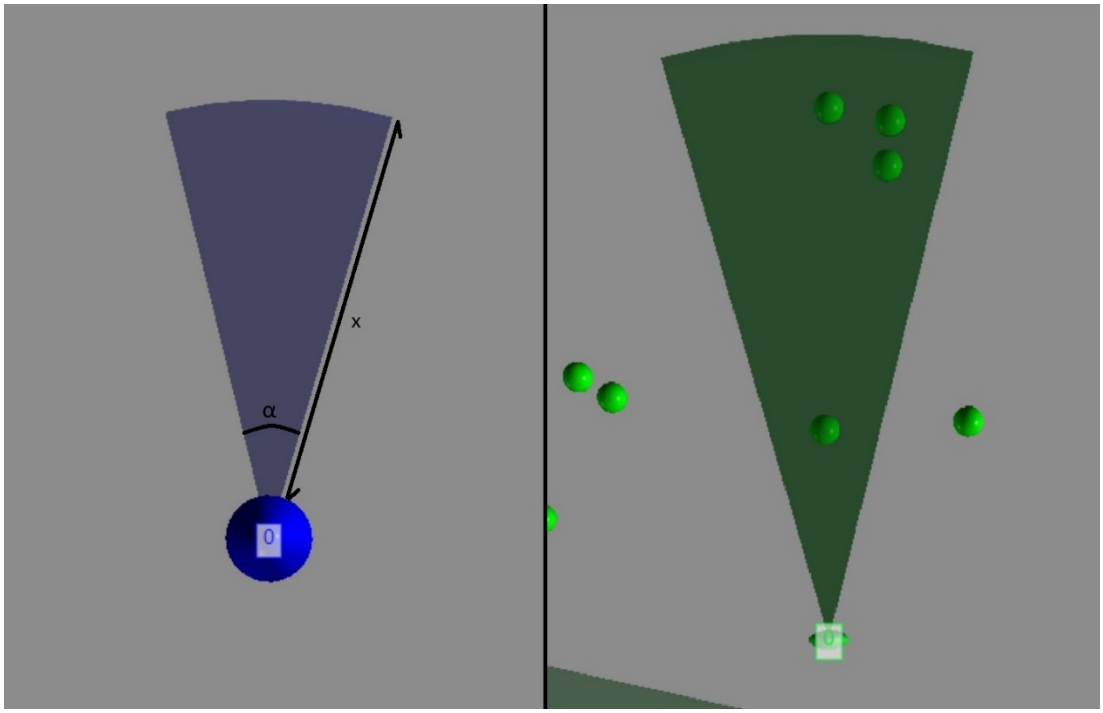
- Durability (dayanıklılık): Canlının tüm özelliklerinin bütün olarak ele alınmasıyla hesaplanan bir değerdir. Canlı ne kadar hızlı ve sağlam bağışıklığa sahipse o kadar dayanıklı olur. Dayanıklılık canlının hastalıklarla mücadelesinde rol oynar.

Günü başarıyla bitirebilen canlılar mutasyon şanslarını denerler. Kullanıcının gerçek zamanlı olarak belirlediği bir olasılık dahilinde canlı üzerinde rastgele bir genetik faktör yine kullanıcının belirlediği sınırlar içerisinde değişir. Bu değişim canlının yararına ya da zararına olabilir. Örneğin gün sonunda yemeğe ulaşan bir canlı eğer ki mutasyon geçirirse görüş açısını 90°ye de çıkartabilir, 10°ye de düşürebilir.

Mutasyon veya bir başka nedenle geni değişen canlılar gün sonunda sağ kalabilirlerse filogenetik ağaç üzerine yeni bir dal olarak eklenir.

Gün sonunda canlının kazandığı özellik canlıya sonraki gün yemeğe ulaşmak için avantaj sağlarsa canlının nesli bir gün daha yaşamış olacaktır. Ortamın günlerce çalışması nesilden nesile toplumun genetik gidişatındaki popülasyon dağılımının hesaplanması ve takibi çıkarım yapacağımız grafiklerdir. Mutasyonların olasılıklarının popülasyon üzerindeki etkileri ulaşmak istenen sonuç grafiklerdir. Farklı olasılık ve değişkenlerin seçilimi ne düzeyde etkileyeceğini kontrollü bir ortam içerisinde simüle ederek popülasyon dağılım ilkeleri oluşturulacaktır. ‘Creature’ sınıfı içerisindeki bazı fonksiyonlar ve özellikleri:

- pos(): Canlının hızına bağlı olarak ilerlemesini sağlar
- foodsinSight(): Canlının görüş alanı ve mesafesine göre görüşünde bulunan yiyecekleri listeler
- collide(): Canlının görüşündeki yiyeceklere ulaşma durumunu kontrol eder
- goCloserFood(): Görüş alanının içerisindeki en yakın yiyeceğe gitmeyi hedefler
- searchFood(): Yukarıdaki 3 fonksiyonu takip eden ve kontrol altında tutar.
- mutation(): Canlının olasılıklar dahilinde mutasyona uğramasını ve mutasyonun görselleştirilmesinde etkilidir.



Şekil 3.4. Örnek iki canlı

Şekil 3.4.'de örnek olarak verilen, id:0 olan canlı α görüş açısına sahip ve x uzunluğunda görüş mesafesine sahiptir. Gün sonunda geçirdiği mutasyon ile görüş mesafesini 10 birimden, 30 birime çıkarmıştır. Bu sayede daha uzaktaki yiyecekleri görerek avantaj kazanmıştır.

creature.py > ...

```
1   import vpython as vp
2   from time import *
3   import random
4   from environment import envSizes
5   import mutationFactors
6   import math
7   import fovCalculation
8
9   class Creature:
10      time = 0
11      def __init__(self, idnumber, color=(0,0,1), speed=0.3, vision=30, visionRadius=10, axis=(1,0,0), pos = (0,1,-35)):···
38
39      def unVisible(self):···
43
44      def pos(self):···
57
58      def collide(self,foods):···
66
67      def foodsinSight(self,foods):···
77
78      def goCloserFood(self, food):···
94
95      def searchFood(self,foods):···
118
119     def mutation(self):···
148
```

Şekil 3.5. Creature sınıfı genel görünümü

3.1.2. Çevre ve yiyecek

Çevre projenin kalanı gibi farklı ölçeklere uygun olacak şekilde geliştirilmesi amaçlanmıştır. Çevreyi oluşturan iki faktör bulunmaktadır:

- Canlılar ve Yiyecekler
- Fiziksel veya Viral afetler

3.1.2.1 Canlılar ve yiyecek

Çevreye istenen miktarda, rastgele konumlarda olacak şekilde yiyecek dağıtılır. İstenilen canlı ve yiyecek sayısı kullanıcı tarafından belirlenir. 'Chars' ve 'Foods' sınıfı tarafından canlı topluluğu ve yiyecek topluluğu oluşturulur.

```
main.py > ...
1   import vpython as vp
2   import time
3   import environment
4   import creature
5   import random
6   import copy
7
8   env = environment.Enviroment(environment.envSizes)
9
10  class Chars:
11      chars = {}
12      def __init__(self, number=10):···
17
18      def setPos(self,foodlist):···
21
22      def results(self):···
58
59      def resetPos(self):···
74
75      def endofDay(self,Foods):···
85
86  class Foods:
87      foods = {}
88      def __init__(self,number):···
96
97      def restartFoods(self):···
109
110 dozenChar = Chars(1)
111 dozenFood = Foods(20)
112
```

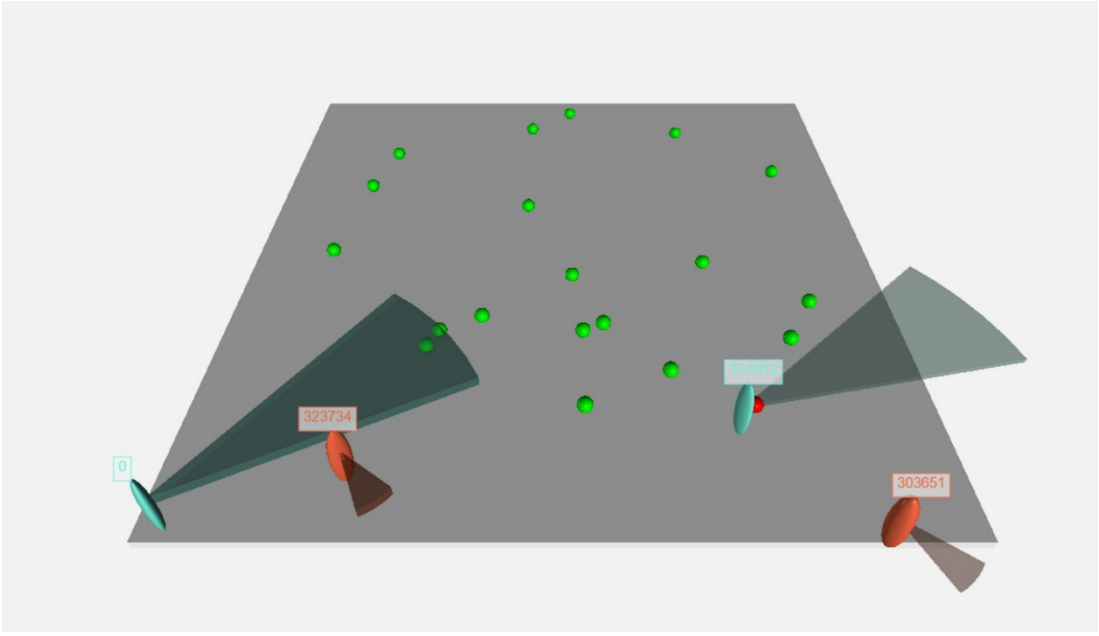
Şekil 3.6. Canlı topluluğu ve yiyecekler

‘Chars’ sınıfı içerisindeki bazı fonksiyonların özellikleri:

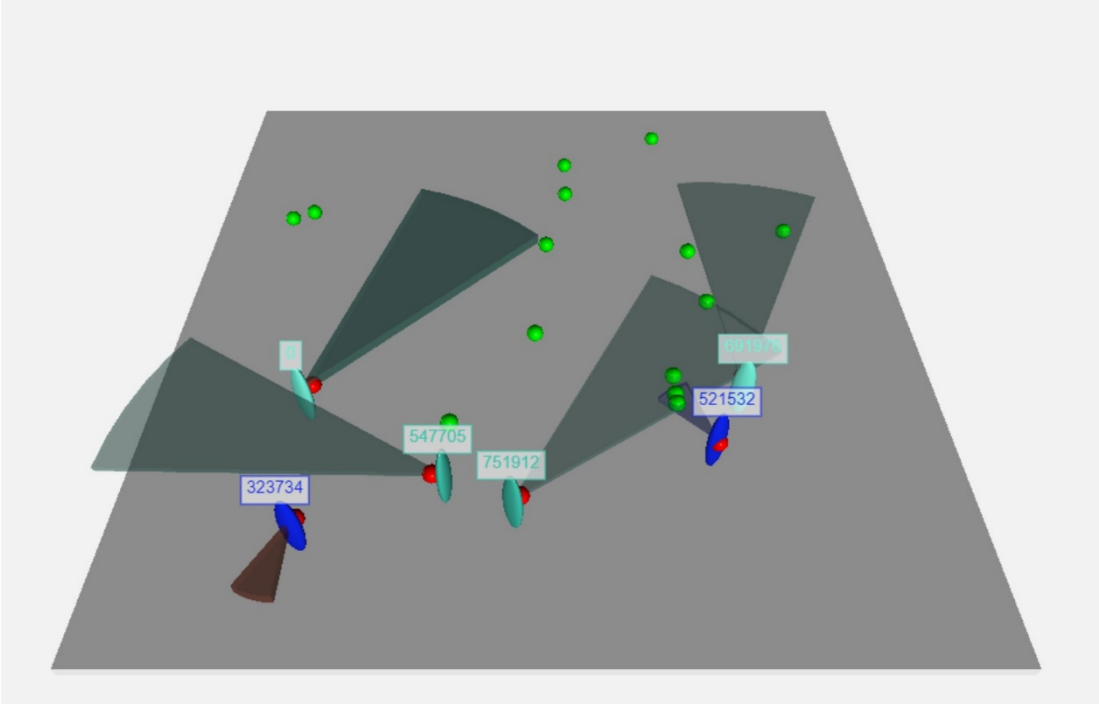
- setPos(): Her canlının gün içerisindeki pozisyonlarını günceller.
- results(): Gün sonunda yaşanan olayları ekrana basar. Mutasyonlar, çoğalmalar vb.
- resetPos(): Yeni günün başlangıcında canlıların başlangıç pozisyonlarına atar.
- endofDay(): Gün sonunda gerçekleşecek olaylar bütününü kontrol eder. Yiyecek bulamayanları silme ve yiyecek bulabilenlerin çoğalmasını sağlar.
    - dailyStats(): Gün sonunda günün kayıtlarını tutar.
    - resetPos(): Canlıların konumlarını yeni gün için sıfırlar.

‘Foods’ sınıfı içerisindeki bazı fonksiyonların özellikleri:

- restartFoods(): Gün sonunda çağrılarak yeni günün yiyeceklerini yerleştirir.



Şekil 3.7. Çevre, Canlılar ve Yiyecekler



Şekil 3.8. Farklı türlerin oluşumu

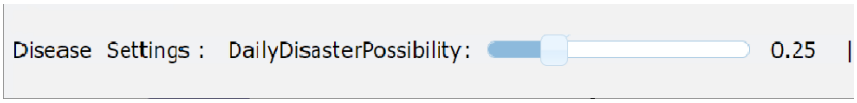
```
0 found food: 0
End of The Day
0: Got Mutation for --visionRadius-- old spec:10 -- new:29.78
0 duplicated 323734
Results:
Winners: [0]
Losers: []
323734 found food: 4
0 found food: 14
End of The Day
0: Got Mutation for --visionRadius-- old spec:29.78 -- new:27.71
323734: Got Mutation for --visionRadius-- old spec:29.78 -- new:6.03
0 duplicated 751912
```

Şekil 3.9. Örnek çıktı

Şekil 3.9'deki çıktıyı incelediğimizde:

- id: 0 olan canlımız gün içerisinde yiyeceği buluyor.
- Gün sonunda mutasyona uğruyor ve görüş mesafesini 10'dan 29.78'e çıkarıyor.
- Canlı çoğalarak id:323734 olan canlıyı oluşturuyor.
- Sonraki gün içerisinde her iki canlı da yiyeceklerini buluyor ve tekrar mutasyona uğruyorlar.

3.1.3. Fiziksel veya viral afetler



Şekil 3.10. Afet olasılığı

Kullanıcı arayüz üzerinde bir sonraki günün afet olasılıklarını belirleyebilmektedir (Şekil 3.10.). Daily Disaster Possibility, sonraki gün afet gerçekleşme olasılığıdır.

Şekil 2.11 Afet türü

Kullanıcı Şekil 3.11 üzerinden gerçekleşecek afetin tür ve şiddetini belirleyebilir. Sayının değeri afetin türünü belirlediği gibi büyüklüğünü de belirler. Örneğin 1.1 seviyesinde yaşanacak afet 1. tür olay %10 büyüklükte demektir. 1.8 seviyesinde bir olay ise 1. tür %80 büyüklük demektir.

Gerçekleşebilecek 3 çeşit afet vardır:

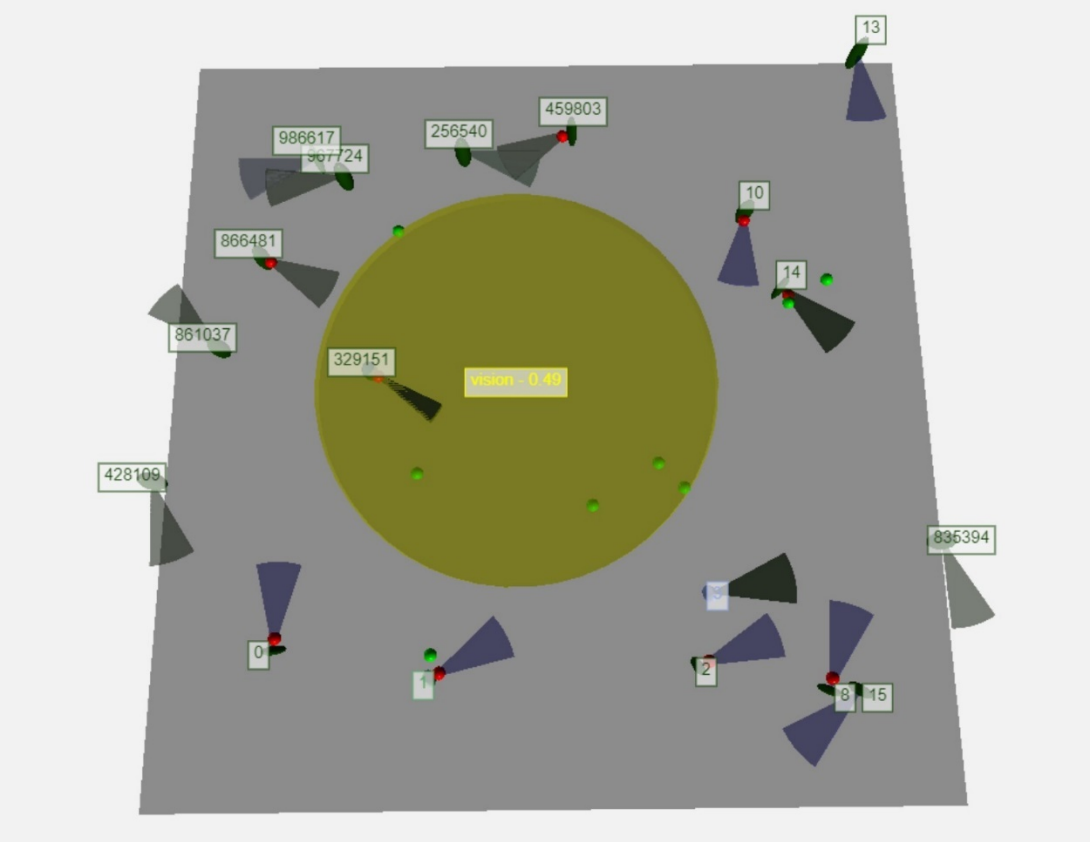
- Temporary Physical Events (geçici fiziksel olaylar) : (1-2)
- Permanent Immune Diseases (kalıcı bağışıklık hastalıkları) : [2-2.8)
- Lethal Diseases (ölümcül hastalıklar) : [2.8 ,3]

3.1.3.1. Geçici fiziksel olaylar

Uygulama rastgele bir konum ve rastgele bir alan belirler. Olayın etkileyeceği nicel değer belirlenir. Bu seçilen değer: görüş açısı, görüş alanı, hız gibi değerler olabilir. Bu afet bölgesi içine giren canlılar seçilmiş özellikten, olayın şiddetine göre bir dezavantaj alırlar.

Bu olaylar gerçek hayatta yaşanabilecek fiziksel zorlukların temsilidir. Örneğin canlının hareketini kısıtlayan bir bölgeye girmesi gibi veya görüş mesafesini düşüren bir ortamda bulunması gibi.

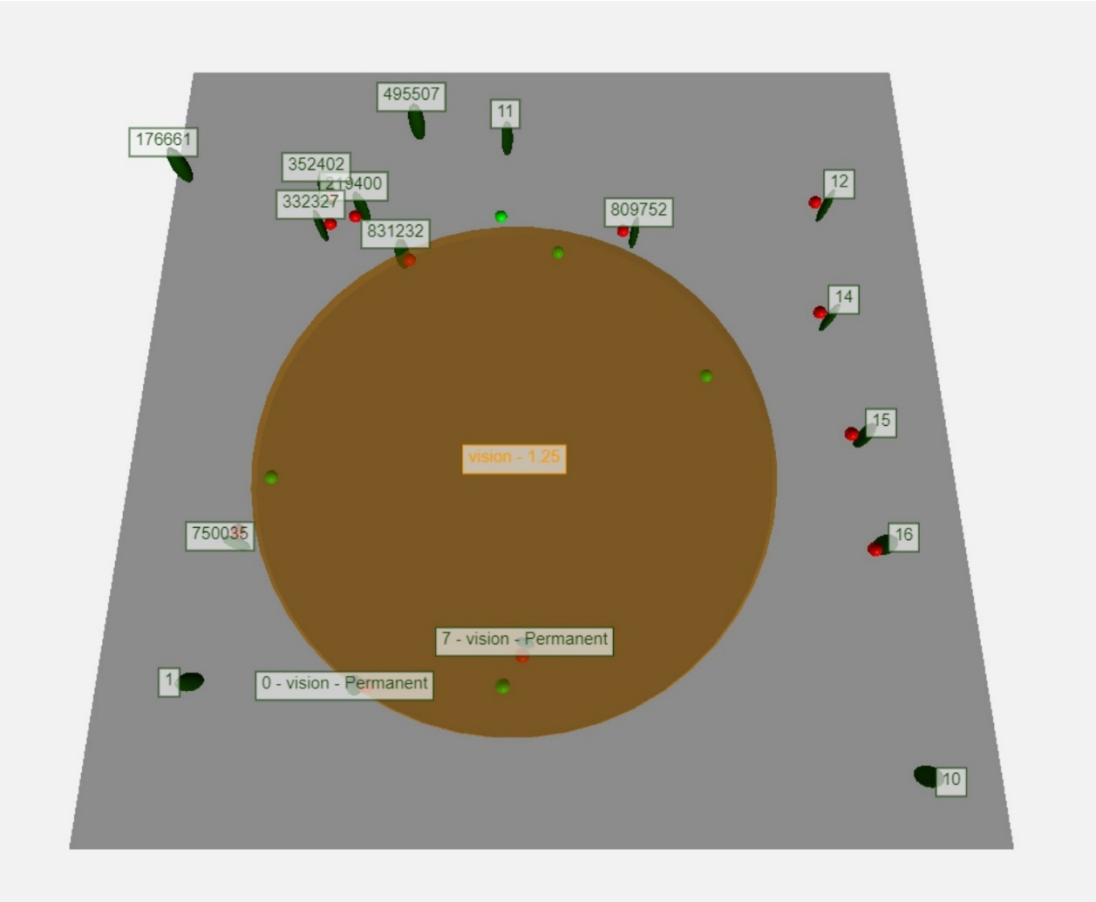
Bu olaylar sadece geçiçi olarak olay ufku içerisinde gerçekleşmektedir. Olay sınırları dışına çıkınca canlı eski kabiliyetlerine geri kavuşur.



Şekil 3.12. Geçici Fiziksel olay

Şekil 3.12.'de örnek bir afet görülmektedir. Oluşan afet görüş açısını %49 düşürmektedir. Afet içerisindeki 329151 id'li canlının görüş açısını yarıya düşürmüştür.

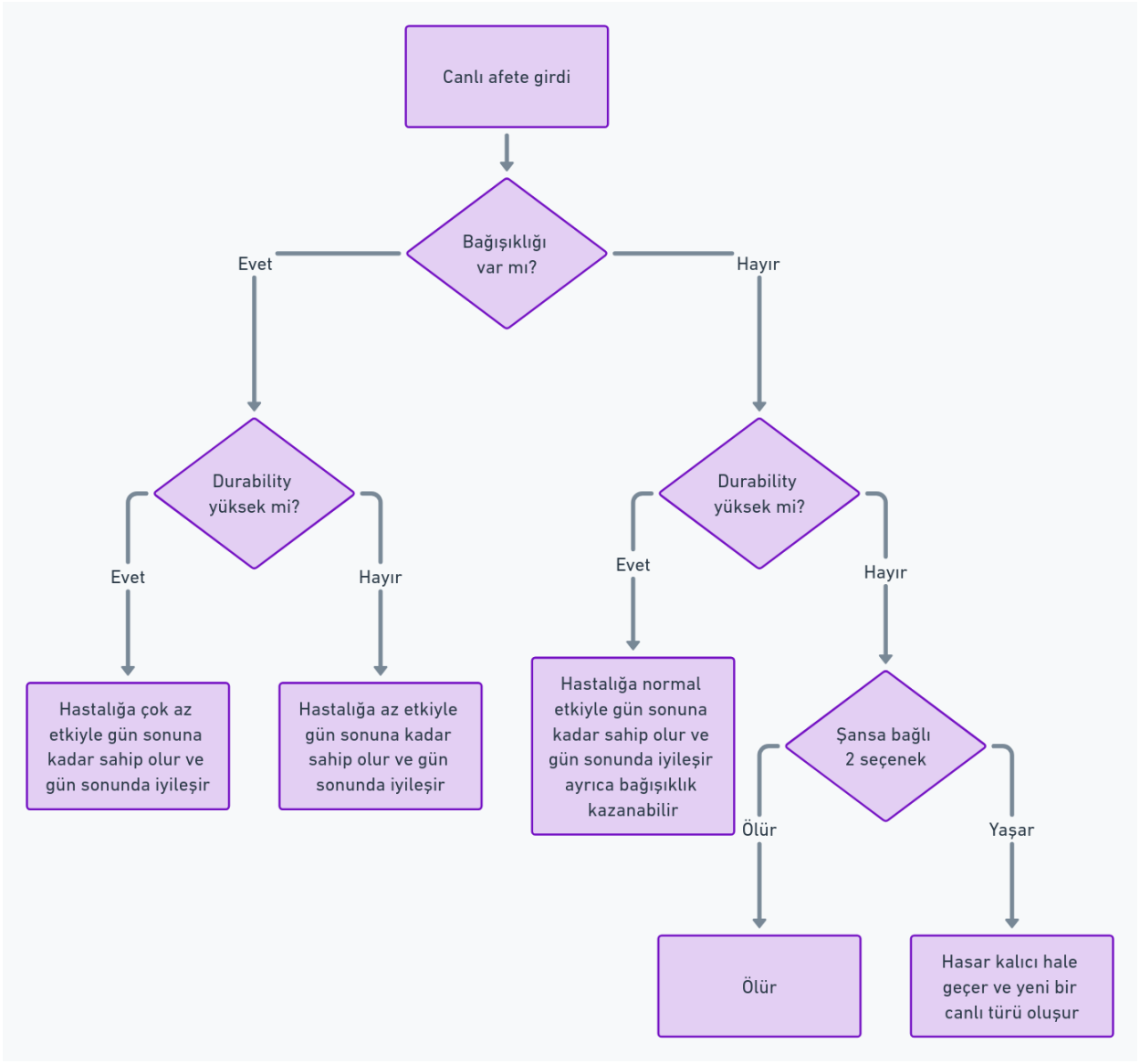
3.1.3.2. Kalıcı bağışıklık hastalıkları



Şekil 3.13. Kalıcı bağışıklık hastalıkları

Bu olay çeşidi viral hastalıkları temsil eder. Rastgele bir virüs, virüsün etki ettiği fiziksel bir özellik ve hastalığın gücü belirlenir. Bu hastalığa dokunan canlılar, canlının bağışıklık ve dayanıklılığına bağlı olarak çeşitli şekillerde atlatabilir:

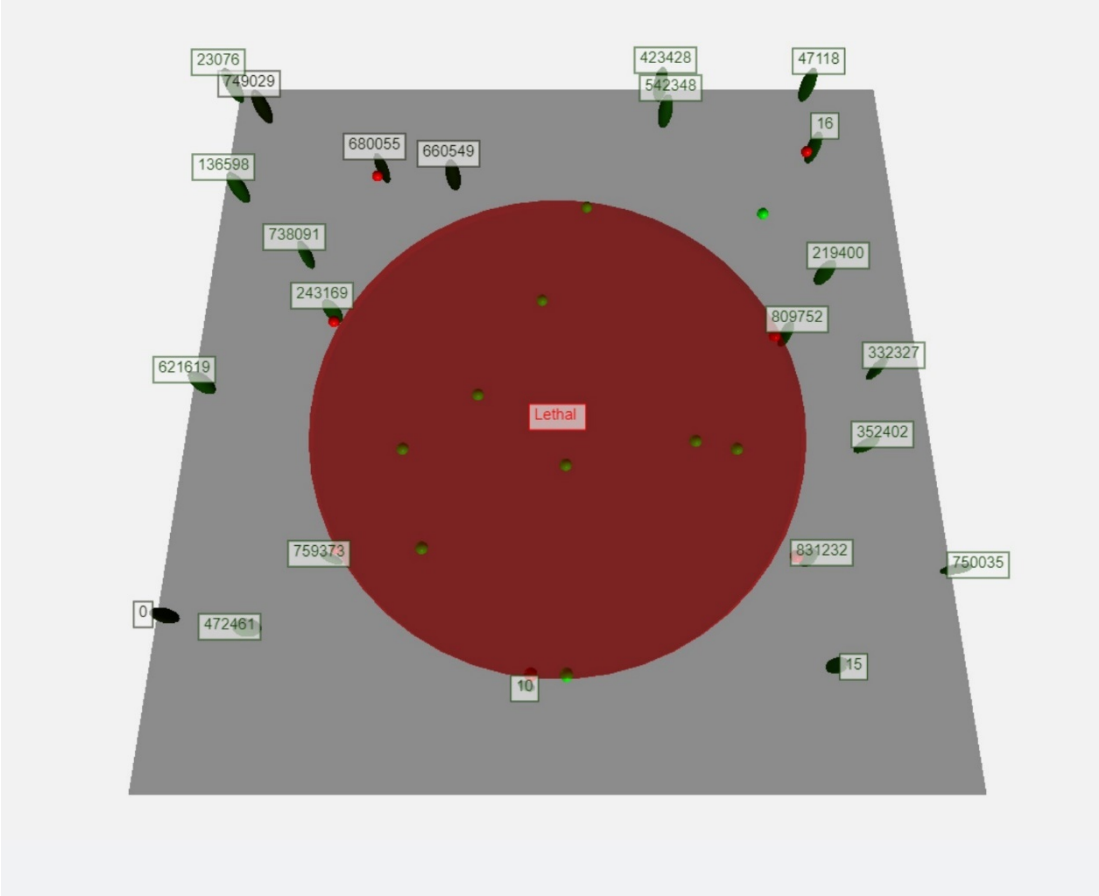
- Bağışıklığı var ve dayanıklı hastalıktan iyi ise, hastalığı çok az etkiyle atlatır.
- Bağışıklığı var ve dayanıklı biri değilse, hastalığı az etkiyle atlatır.
- Bağışıklığı yok ve dayanıklı ise, hastalığı normal bir şekilde atlatır ve bağışıklık kazanır.
- Bağışıklığı yok ve dayanıklı biri değilse, ölebilir veya kalıcı hasar alabilir.



Şekil 3.14. Bağışıklık takibi UML diyagramı

3.1.3.3. Ölümcül hastalıklar

Bu olay ölümcül hastalıkları simüle eder. Bu afet oluşturulduğunda dokunan canlı direkt ölür.



Şekil 3.15. Ölümcül Afetler

3.2. Verilerin toplanması ve grafikler

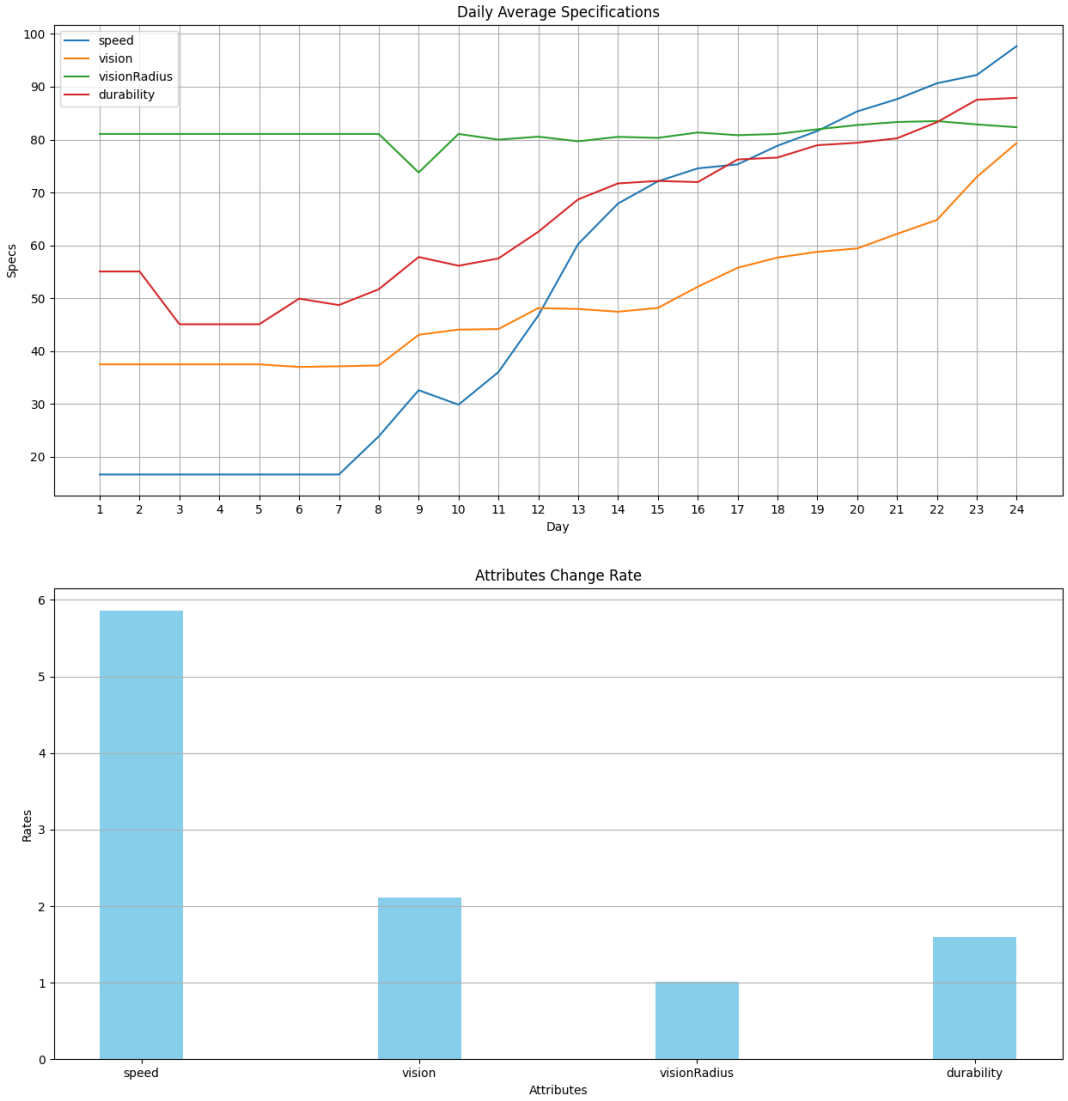
Kullanıcıya sunulmak üzere günlük olarak veriler dailysStats() fonksiyonu tarafından JSON olarak kaydedilir ve deney sonunda sunulmak üzere ayrıştırılır.

```
def dailyStats(self, winners,lenLosers):

    if len(winners):
        sumS = {name: 0 for name in list(self.chars[winners[0]].genomes.keys())}
        for charId in winners:
            for isim in list(sumS.keys()):
                if isim == "immunity":
                    sumS[isim] +=len(self.chars[charId].genomes[isim])
                else:
                    sumS[isim] +=self.chars[charId].genomes[isim]


        self.EoDStats[day] = [values/len(winners) for values in list(sumS.values())]
        self.population[day] = [(len(winners)+lenLosers),len(winners),lenLosers,isDisasterInDay]
        Veri["Day"].append(day)
        Veri["Population"].append(len(winners)+lenLosers)
        Veri["Winners"].append(len(winners))
        Veri["Losers"].append(lenLosers)
        Veri["DisasterType"].append(isDisasterInDay)
        Veri["speed"].append(sumS["speed"]/len(winners))
        Veri["vision"].append(sumS["vision"]/len(winners))
        Veri["visionRadius"].append(sumS["visionRadius"]/len(winners))
        Veri["immunity"].append(sumS["immunity"]/len(winners))
        Veri["durability"].append(sumS["durability"]/len(winners))
        Veri["mutationRate"].append(mutationFactors.mutationProbability)
```

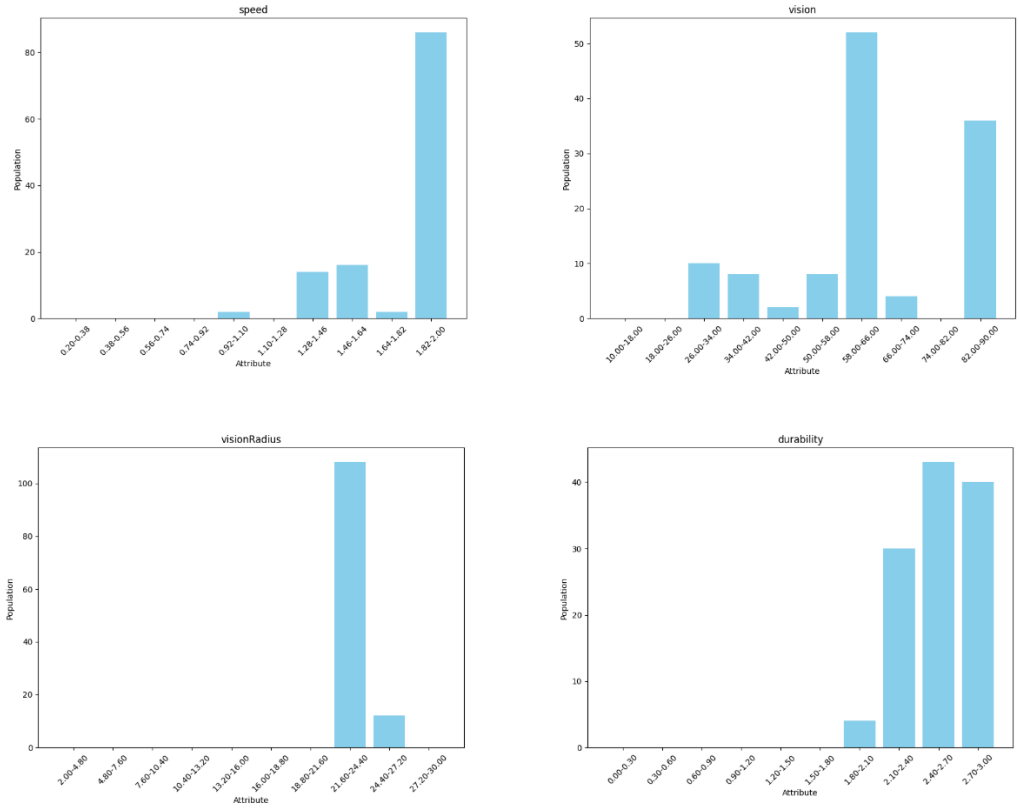
Şekil 3.16. Verilerin kaydedilmesi

Gün sonunda JSON olarak tutulan değerler:

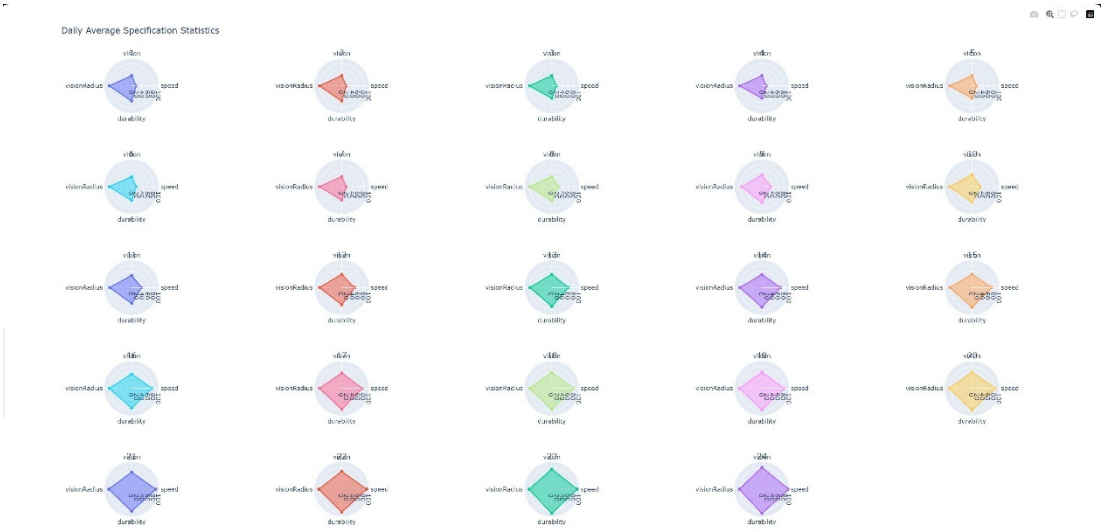
- Gün
- Nüfus
- Yemek bulan canlı sayısı
- Yemek bulamayan canlı sayısı
- Gün içinde yaşanan afetin türü
- Popülasyonun ortalama hız değeri
- Popülasyonun ortalama görüş açısı değeri
- Popülasyonun ortalama görüş mesafesi değeri
- Popülasyonun ortalama bağışıklık değeri
- Popülasyonun ortalama dayanıklılık değeri
- Gün sonundaki mutasyon oranı



Şekil 3.17. Popülasyon grafikleri



Şekil 3.18. Dağılım grafikleri



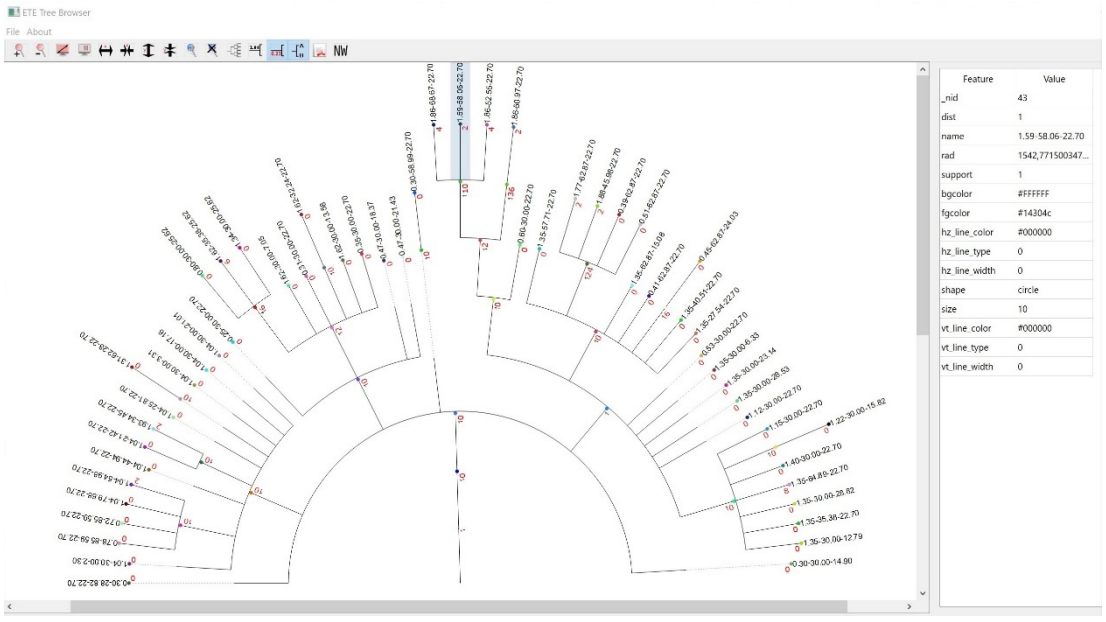
Şekil 3.19. Radar grafikleri

Deney sonunda bu verilerden:

- Canlıların özelliklerinin popülasyon içindeki değişiminin günlük takibi (Şekil 3.17.)
- Her özelliğin deney sonuna kadarki değişim miktarı (Şekil 3.18.)
- Özelliklerin popülasyon içerisindeki dağılımı grafiklendirilmiştir. (Şekil 3.19.)

Herhangi bir canlı gün içinde veya gün sonunda kalıcı genetik değişikliğe uğrarsa yeni bir tür olarak filogenetik ağaca kaydolur.

Filogenetik ağaç genlerin takibini yapmamızı sağlar. Değişen yeni genin hangi genden evrildiği filogenetik ağaçta kaydolur.



Şekil 3.20. Filogenetik ağaç

3.3. Yapay Sinir Ağlarıyla Entegrasyon

Yapay sinir ağlarını entegre etmek adına, canlıların geçmiş davranışları JSON olarak kaydedildi. Daha sonra, bu veri setini kullanarak bir yapay zeka modeli oluşturuldu. Giriş katmanı canlıların özelliklerini, çıkış katmanı ise alması gereken aksiyonları temsil etmektedir.

3.3.1. Veri Seti Oluşturma

Canlıların geçmiş davranışlarını temsil eden bir veri seti oluşturuyor. Bu veri seti, canlıların özelliklerini içeriyor; örneğin, hız, görüş açısı ve görüş mesafesi. Ayrıca, bu özelliklere bağlı olarak alması gereken aksiyonları hesaplıyor.

```
data = json.loads(json_data)
df = pd.DataFrame(data)

# Özellikler ve hedef değişkenler
X = df[['Day', 'mutationRate']]
y = df.drop(columns=['Day', 'mutationRate'])

data = json.loads(json_data)
df = pd.DataFrame(data)
```

Şekil 3.21. Veri seti oluşturma

3.3.2. Modelin Eğitimi

Verileri eğitim ve test setlerine ayırarak modeli eğitmeye başlayalım. Random Forest Regressor kullanarak tüm hedef değişkenler için bir regresyon modeli oluşturuyoruz.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_squared_error

# Eğitim ve test setlerine ayırma
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Multi-output regressor kullanarak modeli oluşturma ve eğitme
model = MultiOutputRegressor(RandomForestRegressor(random_state=42))
model.fit(X_train, y_train)

# Test seti ile modelin doğruluğunu kontrol etme
predictions = model.predict(X_test)
mse = mean_squared_error(y_test, predictions)
print(f'Mean Squared Error: {mse}')

# Tahminleri gösterme
print(predictions)
```

Şekil 3.22 Modelin eğitilmesi

3.3.3. Modelin Canlılara Uygulanması

Modeli kullanarak yeni bir day ve mutation rate verildiğinde diğer değerleri tahmin ediyoruz.

```
# Yeni bir veri seti ile tahmin yapma
new_data = pd.DataFrame({
    'Day': [2],
    'mutationRate': [0.21]
})

new_predictions = model.predict(new_data)
print(new_predictions)
```

Şekil 3.23. Modelin uygulanması

BÖLÜM 4. SONUÇLAR

Simülasyon ortamı çalıştırıldığında istenen ortam değişkenlerine göre bir yaşam döngüsü başlatmaktadır. Simülasyon istenen süre dahilinde ortamı kontrol etmekte ve canlıların animasyonlarını göstermektedir. Simülasyonun sonunda verilen ortam değişkenlerinin canlılar üzerindeki etkilerini görmekteyiz. Üreyen canlı sayıları, canlıların ortalama; hızları, görüş açıları, görüş mesafelerinin süreç içerisinde değişimlerin grafikleri çıkartılmaktadır. Canlıların gen havuzları ete3 aracılığıyla takip edilmiş ve genotipler filogenetik ağaca kaydedilmiştir. Popülasyon verilerinden yorumlar çıkartılmış ve yine aynı verilerle yapay zeka modeli oluşturulmuştur.

KAYNAKLAR

[1] University of California at Berkeley. Natural Selection. (June, 2020). Alındığı Yer: University of California at Berkeley

[2] P. Arnold. Examples Of Natural Selection: A Look At Natural Selection In Action. Alındığı Yer: Brighthub

[3] Husbands, P., Harvey, I., Cliff, D., & Miller, G. (1997). Artificial Evolution: A New Path for Artificial Intelligence? *Brain and Cognition*, 34, 130-159.

[4] Jenkins, B. K., & Tanguay, A. R. (1995). Handbook of neural computing and neural networks. Boston: MIT Press.

[5] Dong, J., & Hu, S. (1997). The progress and prospects of neural network research. Information and Control, 26(5), 360–368.

[6] Brownlee, J. (2018). What is the Difference Between a Batch and an Epoch in a Neural Network. Machine Learning Mastery, 20.

[7] Alber, M., Lapuschkin, S., Seegerer, P., Hägele, M., Schütt, K. T., Montavon, G., ... & Kindermans, P. J. (2019). iNNvestigate neural networks!. J. Mach. Learn. Res., 20(93), 1-8.

[8] Frank, S. A. (2013). Natural selection. VII. History and interpretation of kin selection theory. Journal of Evolutionary Biology, 26(6), 1151-1184.

[9] SEGAL, Mark R. Machine learning benchmarks and random forest regression. 2004.

[10] EL MRABET, Zakaria, et al. Random forest regressor-based approach for detecting fault location and duration in power systems. Sensors, 2022, 22.2: 458.

ÖZGEÇMİŞ

Furkan Liman, 14.11.2000 de Niğde'de doğdu. İlk, orta ve lise eğitimini Niğde'de tamamladı. 2019 yılında Niğde Yunus Emre Anadolu Lisesi'nden mezun oldu, Bilgisayar Bölümü'nden mezun oldu. 2020 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü'nü kazandı.

BSM 498 BİTİRME ÇALIŞMASI
DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI

KONU : YAPAY ZEKA İLE EVRİMSEL SEÇİLİM SİMÜLASYONU
ÖĞRENCİLER (Öğrenci No/AD/SOYAD): G201210007/FURKAN/LİMAN

Değerlendirme Konusu	İstenenler	Not Aralığı	Not
Yazılı Çalışma			
Çalışma klavuza uygun olarak hazırlanmış mı?	x	0-5	
Teknik Yönden			
Problemin tanımı yapılmış mı?	x	0-5	
Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı?			
Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı?			
Yazılımın gereksinim listesi oluşturulmuş mu?			
Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı?			
Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi?			
UML ile modelleme yapılmış mı?			
Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.)			
Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı?			
Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı?			
Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı?			
Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)?			
Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı?			
Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi?			
Yazılımın sızma testi yapılmış mı?			
Performans testi yapılmış mı?			
Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı?			
Yapılan işlerin zorluk derecesi?	x	0-25	
Sözlü Sınav			
Yapılan sunum başarılı mı?	x	0-5	
Soruları yanıtlama yetkinliği?	x	0-20	
Devam Durumu			
Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?	x	0-5	
Diğer Maddeler			
Toplam			

DANIŞMAN (JÜRİ ADINA): DR. ÖĞR. ÜYESİ SERAP ÇAKAR KAMAN
DANIŞMAN İMZASI: