Bilgisayar Mühendisliği

I.Sınıf Yıl Sonu Algoritma ve İleri Programlama Dersi Projesi

Linear Regression

Üyeler

M. Batuhan Ceylan **Berkay Yıldız**

Giriş

Bu projede csv dosyalarından emlak verileri okuyarak onlar üzerinde sıralama, filtreleme ve fiyat tahmini yapma gibi işlemler yapan bir program yazdık. Fiyat tahminleri için iki farklı method sunduk ve bunları ayrıca kıyasladık.

Bu belgede en başta programın temel işlevlerini sergilemek için yazdığımız temel fonksiyonları başlıklar halinde inceliyeceksiniz ardından programın kullanıcı arayüzünü ve işlevlerini gözden geçireceksiniz ve son bölümde kullandığımız iki farklı fiyat tahmini metodunun kıyasına göz atacaksınız.

İşlevler İçin Temel Fonksiyonlar

Bu başlıkta programın kullanıcıya sunulacak temel işlevlerini sağlayan arka plandaki temel fonksiyonları gözden geçireceğiz.

Verilerinin Saklanması

Tekil olarak bir evi saklamak için evlerin genel kriterlerini içeren bir struct yapısı kullandık. Evlerin tamamını saklamak için ise normal bir link list ve link listle üst üste gelen değerlerin tutulduğu baştan sabit boyutlu adres dizisi olan hash table çözümlerini bir arada kullandık. Bunun için ev struct yapımız kriterlerin yanında üç farklı ev pointerı verisi daha ekledik. Bunlar nextHouse, nextHouse_by_id ve nextHouse_by_neighbor şeklinde adlara sahip.

Tekil Olarak Evlerin Tutulması

Program boyunca sürekli olarak elimizdeki ev listesini kırpacağız, flitreleyeceğiz veya sıralayacağız ve bir işlevi gerçekleştirdiğimizde elimizdeki son liste gereksiz bir şeye dönüşecek. Hali hazırda zaten csv dosyalarından gelecek veri miktarını bilmediğimizden ama daha çok da bu listelerin sürekli yeniden kullanımından dolayı hafızada herhangi fazladan alan işgal etmemek ve ev verilerini kopyalamak ile uğraşmamak adına her ev için en başta dinamik olarak bir alan yarattık ve adreslerini onları farklı şekillerde erişebileceğimiz şekilde hash table lara grupladık. İşin sonunda tüm programda sadece program başında erişilen bir fonksiyonda bir tane malloc kullanarak programı tamamladık.

Hash Tablelar ve Link List

Her evi dinamik olarak tuttuğumuz için onları ev başına sadece bir adres alanı daha harcayarak farklı listelerde / hash table larda tutabilme şansımız bulunuyordu ve ileride kolaylık sağlamak için her csv dosyası için iki farklı hash table yaratmaya karar verdik. Birisine evleri id değerlerinden hash üreterek yerleştirdik diğerinde ise hash değerlerini evlerin mahalle kriterinden ürettik. Id üzerinden ilerleyen hash table ın ilk amacı tüm ev adreslerini herhangi bir null hash barındırmadan bir arada tutmak fakat bir yandan da direk olarak id ile bir ev verisine erişmek istediğimizde maksimum ev sayısı / hash table uzunluğu adımda ev adresini elde edebilmek. Mahalle üzerinden ilerleyen hash table ise aslında her mahallenin bir link listini elimizde barındırmak için yarattığımız bir şey, buna benzerlik yöntemi ile ev tahmini yaparken fazlasıyla ihtiyacımız olduğundan böyle bir şeyi elimizde barındırmak büyük bir kolaylık sağlıyor bize, ayrıca yine direkt olarak bu oluşturma sebebimiz olmasa da farklı yerlerde var oluşundan faydalandık.

Hash table ların yanında program içinde evlerle çalışmak için ise evler bir adres değeri daha barındırıyorlar fakat bu adres değeri hash table larınki gibi programın başından sonuna aynı kalmıyor. Bu adresi herhangi bir sebepten evler arası bir liste yaratmak istediğimizde evleri birbirine bağlamak için kullanıyoruz.

Verilerin Okunması

Ev verilerini okuyan fonksiyon iki hash table, dosya yolu ve veri tipi (test/öğrenme) bilgisini alıyor. En başta dosyayı açıp onu satır satır okuyor. Okuduğumuz her satırı bize bir ev adresi döndüren fonksiyona gönderiyoruz. Bu fonksiyon dinamik olarak bir ev structu için alan oluşturuyor ve ona gönderilen satırın test verisi veya öğrenme verisi olma durumunu hesaba katarak satırı parçalıyor ve onu oluşturduğu alana yazıp alanın adresini döndürüyor. Ardından bu adresi istediğimiz bir adresi istediğimiz bir hash table a yerleştiren bir fonksiyona gönderiyoruz. Bu fonksiyon yerleştirme ölçütüne göre evden kriteri alıyor, hash değerini buluyor ve tabloda gezinerek doğru yere yerleştiriyor. Okuma fonksiyonunun çalışması bitince o csv dosyası için id ve mahalle değerlerini baz alan iki hash table doldurulmuş oluyor.

```
int read_house_from_file(char* filename, House * hById[], House * hByN[],
int file_type);
```

Ev verilerini okuyan fonksiyon

Verilere Erişimi Kolaylaştırmak

Verilere erişimi kolaylaştırmak, fonksiyonları kısa ve pratik tutmak adına belli fonksiyonlar yazdık ve tanımlar oluşturduk.

Ön Compiler Tanımları

Kriterleri int formatında fonksiyonlara göndermek için ön compiler tanımları ile onları numaralandırdık. Aynı şekilde ileride kullanacağımız sıralama ve bastırma fonksiyonları için de ön tanımlardan faydalandık.

Global Hash Table ve Dosya Yolları

Programda verileri tuttuğumuz hash table ların statik dizi kısmı program başında sabit bir büyüklükle oluşmaktadır. Bu şekilde programda bu tablolara dolayısıyla verilere erişim sağlamak global olarak bu tablolar üzerinden mümkündür. Yine dosya yolları global değişkenler olarak tanımlanmıştır.

Kritere Göre Veri Döndürme Fonksiyonları

Fonksiyonlarda switch yapılarından kaçınmak için evleri ve ön compiler tanımlarını argüman alan ve istenilen kriter verisini döndüren fonksiyonlar oluşturduk, bunu string, int ve pointer geri döndüren biçimlerde üç farklı şekilde yaptık. Ayrıca evler ile kriterlerin kombinasyonlarından oluşan vere ihtiyacımız olan yerlerde de yeni ön compiler tanımları oluşturarak bu fonksiyona yeni özellik olarak bunları ekleyip onun üzerinden kullandık.

Ayrıca bir id değerine göre id değerlerinden oluşan hash table içinde hızlıca aranan evi bulup geri döndüren bir fonksiyon da oluşturduk. Ve bunların dışında bir ev için belli bir kriterin ortalamasını bulan ve mutfak kriteri için harf-sayı dönüşümleri sağlayan fonksiyonları da pratik olması açısından tek başına fonksiyonlar olarak yazdık.

```
char * ghc_s (House * house, int criter_name) {
 switch (criter_name)
   {
   case STREET:
     return house->street;
     break;
   case NEIGHBORHOOD:
     return house->neighborhood;
     break;
   default:
     break;
   }
}
```

Alınan parametreyi string olarak döndüren fonksiyon

```
int ghc_i (House * house, int criter_name);
House * ghc_p (House * house, int criter_name);
```

Parametreleri int ve House pointer'ı olarak döndüren fonksiyonların prototipleri

Verilerin Manipülasyonu

Verileri işlevler içinde sıralamak, sınırlamak gibi işlevler için belli fonksiyonlar tanımladık.

Bağlı Liste Oluşumu

Program fonksiyonların kolay işlemesi adına (kırpma, sıralama gibi işlemlerin kolay olması) fonksiyonlar bağlı listelerle çalışıyor. Gerektiği zaman istenilen bağlı listeden bir bağlı liste yaratması için iki farklı fonksiyon kullanıyoruz. Birincisi tüm bir bağlı listeyi düz ediyor, bunu yaparken idler ile oluşturulan tablo kullanılıyor. İkinci olarak belli bir mahalledeki tüm evleri bir listeye toplamak için mahalleler ile oluşturulan tablo kullanılıyor. Bu fonksiyonlar hash table ları argüman alan fonksiyonlar tarafından kullanılıyor.

```
House* linearise_hash_table (House * ht[], int hash_type, int *
lenght);

House* get_neighborhoods(House * house, House * houses[], int *
lenght);
```

Bağlı listeden bağlı liste yaratan fonksiyonlar

Verilerin Sıralanması

Verileri bağlı listelerde sıralamak için en uygun çözüm olarak merge sort algoritmasını uygun gördük. Fonksiyon isimlerinde karışıklık yaşanmaması için sıralama fonksiyonumuzu merge sort fonksiyonu çağıracak şekilde ayarladık

Verilerin Limitlenmesi

Adresi gönderilen bir bağlı listeyi belli bir kriter için verilen min max değerlerine göre limitleyen ve yine kendisine gönderilen adrese yeni listenin uzunluğunu yazan bir fonksiyon oluşturduk.

```
//İstenilen alt üst değere göre verilen bağlı listeyi kırpar, NON
kullanılarak min veya max belirlenmeyebilir
void limit_houses(House** houses_head, int criter_name, int min, int
max, int * new_lenght)
```

Bağlı listeyi limitleyen fonksiyon

Program Akışı Sırasında Girdi Çıktı

Program akışı sırasında kullanıcıdan alınacak veriler ve kullanıcıya gerek ekrana bastırılacak gerek dosyaya yazılacak veriler için belirli fonksiyonlar oluşturduk

Kullanıcıdan Girdi Alma

Argüman Girdisi

Program dosya yollarını çağırılırken argüman olarak (run -> executable öğrenme.csv test.csv şeklinde) alabiliyor.

Konsoldan Veri Alma

Program akışı sırasında integer char ve string cinsinden okumalar yapılıyor. İnteger ve char cinsinden okumalarda istenilen spesifik girdiler dışında girdi alınmaması için özel okuma yapıp okunanı geri döndüren fonksiyonlar oluşturduk.

Ekrana Bastırma

Ekrana bastırma işi tekil bir ev için veya bir ev dizisi için sürekli olarak gerçekleştiğinden bunu kolaylaştıran bir fonksiyon yazdık. Fonksiyon tek bir ev veya bir bağlı listenin başı olabilecek bir ev adresi, bastırma türü ve maksimum bastırılacak ev sayısı değerlerini alıyor. Bastırma biçimi olarak çoklu, başlıklı, başlıksız gibi seçenekler üretip bunları ön compiler ile tanımladık, yine limitsiz bastırma için de bir tanım gerçekleştirdik.

```
void print_house(House * house, int style, int limit);
```

Evleri bastırma fonksiyonu

Dosyaya Yazma

Dosyaya yazma için ekrana bastırmaya benzer olarak ev adresi, dosya yolu ve bastırma limiti alan ve istenen dosyaya bastıran bir fonksiyon tanımladık.

```
void write_house_to_file(House* head, char* filename, int limit);
```

Evleri dosyaya bastırma fonksiyonu

Kullanıcı Arayüzü ve İşlevler

Programın kullanıcı arayüzünü inceleyeceğiz, kullanıcıya sunulan işlevlerin sonuç üretmek için temel fonksiyonları ve bazı özel fonksiyonları nasıl kullanıklarını inceleyeceğiz. Bahsi geçen özel fonksiyonların bir kısmı işlev açıklanırken bir kısmı da Modelleme bölümünde açıklanacak. Yine

Menüler Arası Geçiş Sistemi

Main içerisinde menü işlevleri benzerlik gösterdiğinden kullandığımız menüleri ön derleyici üzerinden numaralandırdık. Ardından menü, alt menü, mini menü gibi değişkenleri oluşturduk. Her menü yaratmak istediğimiz yerde o yerin menü değişkeni çıkış değeri almadıkça dönmeye devam edecek bir döngü ve menü değişkeninin değerlerini göre menüye giriş sağlayacak sorgular gerçekleştirdik. Aşağıda bahsedilen tüm *menü geçişleri*, *seçenek sunma* ve *soru sorma* durumları bu sistem üzerinden, farklı menü değişkenleri ve döngüler kullanılarak gerçekleşmektedir.

Dosya Yollarının Alınması, Kontrol ve Okuma

En başta test ve öğrenme verilerinin argüman yolu ile alınıp alınmadığı main fonksiyonunun aldığı argümanlar üzerinden kontrol ediliyor. Eğer alınmışsa alındığına dair mesaj veriliyor eğer alınmamış ise kullanıcıya elle girme veya bizim ön derleyici ile ön tanımladığımız adresi alıp almak istemediği soruluyor. Elle girilmesi durumunda uzantı kontrol fonksiyonu ile kontrol sağlanıyor ve csv dosyası olmaması durumunda hata veriliyor ardından otomatik olarak ön tanımlı adres seçiliyor. Bu işlem her iki dosya için de (test/öğrenme) gerçekleşiyor.

Ardından dosyalar okuma fonksiyonuna gönderiliyor, fonksiyondan gelen başarılı okuma değerine göre ana menüye devam ediliyor veya hata verip programdan çıkılıyor.

Ana Menü

Ana menü kullanıcıya beş farklı işlev sunuyor. Ana menü döngüsünde işlevlerin sergilendiği kısım da aslında diğerleri gibi menü değişkeni ile geçilen bir menüden oluşuyor. Program başlatılırken bu menü açılıyor ve diğer işlevler işlemlerini sonlandırdıktan sonra buraya dönüş yapıyor. Keza program kullanıcıya sunulan birkaç yer ve hata çıkışları dışında bu menüdeki seçenek ile sonlandırılıyor. Fonksiyonlar Ana Menüye geri dönmeden önce kullanıcıya aynı işlevi tekrarlamak istemediğini soruyorlar genelde, bu soruya verilen cevaba göre

Kriterlere Göre Veri Listeleme

Bu işlevde kullanıcı istediği bir csv dosyasını istediği kadar kriter için alt üst değerler vererek filtreleyip, istediği yönde sıralayıp, o yönde dilediği miktarı dosyaya yazdırabiliyor veya ekrana bastırabiliyor. Bu işlev için sorular eşliğinde:

- İstenilen Grubun Hash Table I Bağlı Listeye Dönüştürülüyor
- İstenilen Kritere Göre Bağlı Liste Sınırlanıyor
- Tekrar İstenmeyene Kadar Bir Üstteki Adım Tekrarlanıyor
- Liste Sıralanıyor
- Maksimum Basılacak Değer Sayısı Kaydediliyor
- Görüntüleme Biçimi Soruluyor
- Dosyaya Basma Seçilirse Dosya Yolu Sorulup Bastırılıyor
- Ekrana Bastırma Seçilirse Ekrana Bastırılıyor

ID si Verilen Evi Bulma

Bu işlev için kullanıcıdan id değeri ve verilen id'nin hangi veri grubunda bulunduğu bilgisi alınıyor. Id ile oluşturulan hash table üzerinden ev döndüren fonksiyon ile ev değeri alınıyor ve ev bastırma fonksiyonu ile bastırılıyor.

Komşu Evleri Listeleme

Bu işlev kullanıcıdan bir id ve evin bulunduğu verisini alıyor. İlk önce id ile ev döndürme fonksiyonu ile evi alıp ardından evin komşularını o veri grubunun mahalle ile oluşturulmuş hash table ından alan fonksiyon ile komşuların listesini alıyor. Görüntüleme biçimine göre dosya yolu sorarak dosyaya basma fonksiyonu ile basıyor veya ekrana basma fonksiyonu kullanıyor.

Kriterlere Göre Gruplandırarak Gösterme

Bu işlev kullanıcıdan aldığı bir kritere göre öğrenme verisini gruplandırıyor. Bunun için direkt olarak hash table ve kriter alan bir fonksiyon çağırıyor. Grupları büyüklükleri ve fiyat ortalamaları ile bastırıyor. Bu fonksiyon tekil bir fonksiyon olarak istendiğinden ve ekrana bastırma şekli yönünden farklı olduğundan diğer temel fonksiyonlardan farklı bir ekrana basma biçimi içeriyor. Mahalle için mahalle hash table ını kullanıp direkt olarak her hash değerindeki evlerin sayısını ve ortalama fiyatını buluyor ve ekrana basıyor.. Diğerleri için id hash table ını alıp sıralıyor ardından baştan sona doğru giderken evlerin fiyatlarını topluyor, değişen değerlerde durarak öncekilerin sayısını ve ortalamalarını ekrana bastırıyor ve toplamı sıfırlıyor.

Fiyat Tahmini Yapma

Bu işlev kullanıcıların doğrusal metod ve benzerlik metodları ile test verisindeki veya herhangi id si verilen bir evin fiyatını tahmin etmelerini sağlıyor. Kullanıcıdan veri grubu, sonuç görüntüleme biçimi (ekrana bastırma / dosyaya yazma), tahmin metodu gibi bilgileri aldıktan sonra tahmin yapılacak metoda göre belirli işlemler gerçekleştirerek sonucu sergiliyor. Burada bahsi geçen metodlar ve işleyişleri Fiyat Tahmin Yöntemleri başlığında incelenecektir.

Fiyat Tahmin Yöntemleri Karşılaştırma

Bu işlev verilmiş olan train datasını kopyalar, içindeki evlerin fiyatlarını iki yöntemle de tahmin eder ve sonuçlarını bastırır. En sonunda da tahminlerin gerçek değere uzaklıkları üzerinden hesaplanan ve karşılaştırma sonucu niteliğinde bir veri yansıtır.

Fiyat Tahmin Yöntemleri

Tahminleri gerçekleştirmek için biri “linear regression” diğeri ise evler arasındaki benzerliğe dayanan iki farklı metod kullandık.

Linear Regression

Linear Regression yöntemi, fiyatları tahmin ederken matris çarpımı yöntemini baz alır. Bu yöntemde tüm veriler X ve Y matrislerine yazılır. X matrisinin her satırında evlerin ‘lotarea’ verisi ve işlemlerde düzgün sonuç çıkması için eklenen 1 sayısı bulunur. Y matrisinde ise aynı evlerin ‘saleprice’ verileri her bir satıra yazılır. Bu matrisler belirli işlemlerden geçerek W isimli parametre matrisi oluşturur. Bu matriste calculate_parameter() fonksiyonu ile oluşturulur. Daha sonra make_prediction() fonksiyonu ile bu matristeki sonuçlar ve test evlerinin ‘lotarea’ parametresi kullanılarak bir çarpım yapılır. Bu çarpımın sonucunda tüm test evleri için tahmini bir fiyat elde edilmiş olur. Bu tahminler doğrusal bir orantıdadır ve bu yöntem de ismini bu sonuçtan gelen XY doğrusu üzerinde bir çizgi olarak ifade edilebilen doğrudan alır.

Benzerlik

Benzerlik yönteminde fiyatı tahmin edilmesi gereken verinin öncelikle komşularını hash table dan getiren fonksiyon ile liste olarak alıyoruz. Ardından sırasıyla alan, yapım yılı ve son olarak üç farklı kalite değerinin ortalamasından oluşan bir değeri ve bunların birimlerine uygun uzaklık değerleri ile elimizdeki listeyi kırpıyoruz. Her kırpmada kalan listenin uzunluğunu kontrol ediyoruz. Eğer liste uzunluğu 10 dan az ise kalan evlerin ortalamasını kritere göre liste ortalaması alma fonksiyonu ile alarak geri döndürüyoruz.

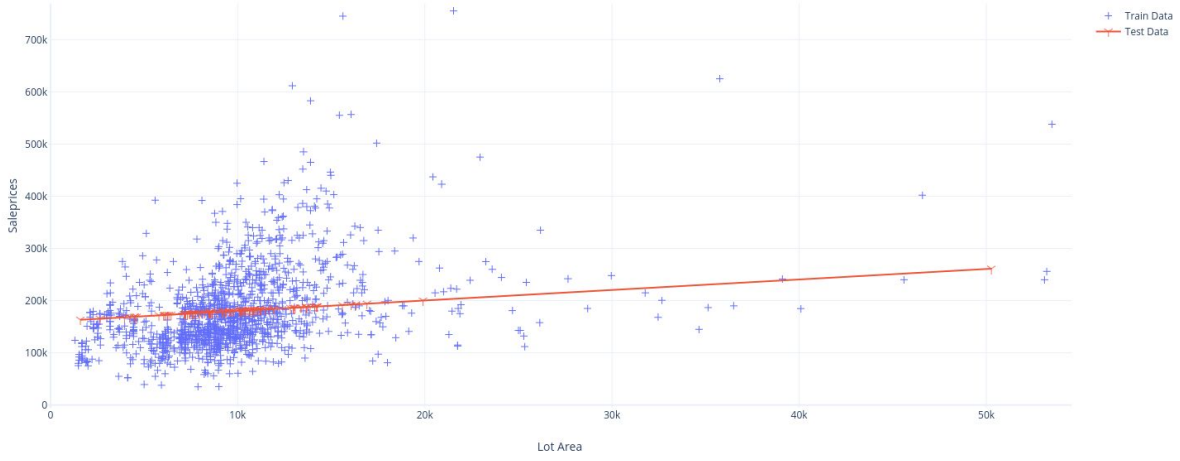
Yöntemler Arasındaki Kıyas

İki yöntem genel olarak ilk bakışta çok farklı tahminler yapmıyor gibi görünmekle birlikte iki yöntem arasındaki kıyası veri olarak sağlamak istedik. Bunun için hali hazırda fiyatı olan öğrenme verilerinin fiyatını her iki yöntemle de tahmin ettik. Ardından iki yöntemin tahminini de gerçek fiyatla kıyasladık ve çıkan farkları kendi aralarında topladık. En sonunda iki yöntemin hata paylarının toplamını yan yana koyduğumuzda matrix verisi iki kata yakın oranda daha fazla hata yapabilir durumda çıktı. Bunun sebebini anlamak için elimizdeki verileri kullanarak Python dilini kullandığımız bir kütüphaneden de faydalanarak programda veri olarak bastırdığımız doğruyu çizdik. Veri grubunun doğruya uzaklığı sebebiyle bu yöntem yüksek ihtimalle verimsiz kalmış gibi görünüyor. Yine daha doğru bir karşılaştırma yapabilmek için benzerlik yöntemi ile zaten verilerin var olduğu bir grupta liste kırparken elimizde son olarak tek ve aynı veri kalma ihtimali olduğu için (kısaca asıl fiyatı geri tahmin fiyatı döndürdüğümüz için) bunları eleyerek bir tahmin daha yaptık fakat bu da pek farklı bir sonuç vermedi.

Kıyas

```
Benzerlik ile sıfır çıkanlar dahil

Ev : 1300

Bernzerlik : 34140400

Matrix : 75273484

Fark : 41133084

Benzerlik ile sıfır çıkanlar dahil edilmeden

Ev : 1305

Bernzerlik : 33878400

Matrix : 70177984

Fark : 36099584
```

Verilerin Grafik Hali



Sonuç

Sonuç olarak benzerlik yönteminin daha iyi olduğunu verilerimizle görmüş olduk.