PROJE ÖZETİ

Makine Öğrenmesi ile Çalışma Verimi Tahminleri Yapan Bir Yazılım

Özet:

Bu projenin amacı *"Yaptığımız aktivitelerde alacağımız verimi daha önceden bilseydik nasıl olurdu? Daha aktiviteye başlamadan hangi aktiviteden ne kadar verim alacağımızı tahmin etmenin bir yolu var mıdır? Eğer var ise tahmin için gereken bilgileri kullanıcıdan nasıl alabiliriz ? Aldığımız verileri nasıl işleriz ? "* gibi sorulara çözümler bulmaktır . Projemizde bu tahminleri yapmak için Makine Öğrenimi algoritmalarından yararlandık. Yaptığımız yazılım daha aktiviteye başlamadan bize psikolojik durumumuz ile ilgili soruları sormaktadır . Bu sorgu değerlerinden yararlanarak iki ayrı tahmin yapar. Yapılan ilk tahmin ,bilgi girdisinde bulunulan ve şartları sağlayan aktivitelerden hangisinden yüzde kaç verim alınacağının tahminidir . Bu tahmin yapılırken regresyon algoritmalarından yararlanılır . İkinci tahmin şekli ise genel aktivite tahminidir. Bu tahmin , kullanıcının bulunduğu psikolojik halde genel olarak hangi aktiviteyi yaptığının tahminidir. Bu tahmin şekli verim odaklı değildir . Sınıflandırma algoritmalarından yararlanılır.

Yukarıda anlatılanlar yaptığımız programın temel amacıydı . Kullanıcının yukarıda anlatılanları kullanabilmesi için bir arayüz gerekmektedir. Yaptığımız program kullanıcıya günlük çalışma programını hazırlayabileceği , psikolojik sorgu değerlerini girebileceği, yaptığı aktiviteler hakkında bilgi alabileceği bir arayüz tasarımı sunmaktadır .

Bütün bu bilgiler ,günlük çalışma programı , duruma bağlı psikolojik sorgu değerleri , güvenlik isteyen bilgilerdir . Yanlış ellere düşmesi durumunda kötü sonuçlar ortaya çıkabilir. Bu sebepten ötürü yaptığımız programda kullanıcı kayıt sistemini geliştirdik. Projemizin bir sonraki versiyonunda kullanıcının bu bilgilerini sakladığı veritabanlarını , şifrelenmiş veritabanı haline getirmeyi düşünmekteyiz . Kayıt sistemi aynı zamanda birden fazla kullanıcıya birden hizmet verme imkanı sunmaktadır.

Anahtar Kelimeler: Makine Öğrenimi , Regresyon , Sınıflandırma , Veritabanı

PROJE PLANI

1. Amaç ve Kapsam:

Bu projeyi yaparken, belirli bir süreçte elde edilen çalışılan konu ve kullanıcının psikolojik durumu gibi değerleri veri kümesi haline getirmeyi (Preproccesing),oluşturulan veri kümesini belirlenen Makine Öğrenmesi algoritmalarına uydurmayı(Fitting),algoritma modelleri arasından en uygun modeli bulmayı(Model Selection) , model aracılığı ile kullanıcıya daha çalışmaya başlamadan o anki girdiği bilgiler durumunda genel olarak hangi aktiviteyi yaptığını ve hangi aktiviteden tahmini olarak ne kadar verim alacağını tahmin etmeyi hedefledik.

Ayrıca olarak bütün bunların kullanımını kolaylaştırmak adına kullanıcının günlük çalışma programını hazırlayabileceği , psikolojik sorgu değerlerini girebileceği bir arayüz tasarlamayı ve bütün bu girilen değerleri saklayabilecek bir veritabanını oluşturmayı hedefledik. Burada ek olarak kullanıcının güvenliği ve çoklu kullanıcı desteği adına kullanıcı kayıt bilgilerinin saklanacağı ayrı bir veritabanı oluşturmayı amaçladık.

2. Yöntem ve Gereçler:

Projemizi yaparken programla dili olarak Python'ını , veritabanı olarak SQLite'i , Makine Öğrenmesi algoritmaları olarak , bazı gözetimli öğrenim(supervised learning) algoritmalarını ve arayüz tasarım için ise PyQT5 paketini kullandık.

Projenin yapımı 3 temel parçadan oluşur. İlk aşama Makine Öğrenmesi kodlarının hazırlanması , ikinci aşama kullanıcı verilerinin SQLite veritabanından kullanılabilir hale getirilmesidir. Üçüncü aşama ise PyQt5 ile kullanıcı arayüzünün tasarlanmasıdır.

3. Kaynaklar :

1. Raschka Sebastian,Python Machine Learning,2015,Packt Publishing (9 Kasım 2018 Tarihinde erişildi)
2. Başer Mustafa ,Python,2018,Dikeyeksen Yayıncılık (9 Kasım 2018 Tarihinde erişildi)
3. https://belgeler.yazbel.com/python-istihza/ (10 Kasım 2018 Tarihinde erişildi)
4. https://en.wikipedia.org/wiki/Conditional_probability (10 Kasım 2018 Tarihinde erişildi)
5. https://en.wikipedia.org/wiki/Bayes%27_theorem (10 Kasım 2018 Tarihinde erişildi)
6. https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes (10 Kasım 2018 Tarihinde erişildi)
7. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (11 Kasım 2018 Tarihinde erişildi)
8. https://scikit-learn.org/stable/modules/neighbors.html (12 Kasım 2018 Tarihinde erişildi)

9. http://madhugnadig.com/articles/machine-learning/2017/03/04/implementing-k-meansclustering-from-scratch-in-python.html (12 Kasım 2018 Tarihinde erişildi)
10. https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/ (13 Kasım 2018 Tarihinde erişildi)
11. https://www.sqlite.org/different.html (14 Kasım 2018 Tarihinde erişildi)
12. https://www.cyberciti.biz/python-tutorials/securely-hash-passwords-in-python/ (14 Kasım 2018 Tarihinde erişildi)
13. https://likegeeks.com/pyqt5-tutorial/ (15 Kasım Tarihinde erişildi)
14. https://stackoverflow.com/questions/685206/how-to-get-a-list-of-column-names (4 Aralık 2018 Tarihinde erişildi)
15. https://stackoverflow.com/questions/7727863/how-to-make-a-cell-in-a-qtablewidget-readonly (4 Aralık 2018 Tarihinde erişildi)
16. https://stackoverflow.com/questions/51619186/pyqt5-qtablewidget-right-click-on-cell-wontspawn-qmenu (5 Aralık 2018 Tarihinde erişildi)
17. https://stackoverflow.com/questions/44264157/pyqt-add-right-click-to-a-widget (8 Aralık 2018 Tarihinde erişildi)
18. https://github.com/RavenKyu/OpenTutorials_PyQt/blob/master/QtFramework/ QtWidgets/ QComboBox/QComboBox_03_model_tableview.py (12 Aralık 2018 Tarihinde erişildi)

4. İş-Zaman Tablosu

30 Ekim 2018 - 8 Kasım 2018: Proje konusunun tespiti, proje adı, sorusu ve amacının belirlenmesi, proje günlüğüne başlanması, yapılacak işlerin listelenmesi.

8-15 Kasım 2018: Proje çalışma yöntemleri, kaynak tarama tespiti, yararlanılacak bilim dalları ve kaynak kişilerin araştırılması.

15-30 Kasım 2018: Çalışmamıza ön hazırlık çalışmalarının yapılması. Proje kodlarının çalışma algoritmasının tasarlanması.

3-15 Aralık 2018: Tasarlanan algoritmaların Python programlama dilinde yazılması .

15-17 Aralık 2018: Temel arayüz tasarımının geliştirilmesi . Oluşturulan arayüz'ün daha hoş bir görünüme getirilmesi.

17-27 Aralık 2018: Proje Raporu, Özeti ve Araştırma Planının Son haline getirilmesi.

27 Aralık 2018 – 04 Ocak 2019: TÜBİTAK başvuru işlemleri için ön rapor yazılması ve başvuru işlemlerinin yapılması.

Projenin Adı :

Makine Öğrenmesi ile Çalışma Verimi Tahminleri Yapan Bir Yazılım

1.Giriş

Bir konu üzerinde çalışırken hepimiz en yüksek verimi almayı isteriz.Yapılan her aktivitenin kalitesi aldığımız verim ile eş değerdir. Aldığımız verim ise bazı değişkenlere bağlıdır. Bu değişkenlerden bazıları göreceli olabilirken bazıları ise herkes için - genel olarak - aynı etkiyi yaratmaktadır . Örneğin aktivite yapılmadan önce aktiviteye duyulan isteklilik aktiviteden alacağımız verimi doğrudan etkileyecektir. Bu durumda da küçük bir miktarda da olsa görecelik içerir. Bu gibi verim ile ilintili değişkenler incelenirse verim hakkında daha iyi bilgiler elde edilecektir.

Projemizde, herhangi aktivitenin üzerinde verimi etkileyecek değişkenler düşünülmüş. Belirlenen değişkenlerin değerlerinden yararlanarak , aktiviteden alınacak verimi yaklaşık olarak bulmanın yolları araştırılmıştır.

1.1 Projenin Amacı

Bu projeyi yaparken, belirli bir süreçte elde edilen çalışılan konu ve kullanıcının psikolojik durumu gibi değerleri veri kümesi haline getirmeyi (Preproccesing),oluşturulan veri kümesini belirlenen Makine Öğrenmesi algoritmalarına uydurmayı(Fitting),algoritma modelleri arasından en uygun modeli bulmayı(Model Selection) , model aracılığı ile kullanıcıya daha çalışmaya başlamadan o anki girdiği bilgiler durumunda genel olarak hangi aktiviteyi yaptığını ve o anki bilgiler durumunda hangi aktiviteden tahmini olarak ne kadar verim alacağını tahmin etmeyi hedefledik.

Ayrıca olarak bütün bunların kullanımını kolaylaştırmak adına kullanıcının günlük çalışma programını hazırlayabileceği , psikolojik sorgu değerlerini girebileceği bir arayüz tasarlamayı ve bütün bu girilen değerleri saklayabilecek bir veritabanını oluşturmayı hedefledik. Burada ek olarak kullanıcının güvenliği ve çoklu kullanıcı desteği adına kullanıcı kayıt bilgilerinin saklanacağı ayrı bir veritabanı oluşturmayı amaçladık.

1.2 Python

Python, **Guido van Rossum** adlı Hollandalı bir programcı tarafından yazılmış bir programlama dilidir. Geliştirilmesine 1990 yılında başlanan Python, geliştirilmeye halen devam edilmektedir. Python programlama dilini seçmemizin sebepleri aşağıdaki gibidir :

- Program geliştirme sürecini kısaltır yani hızlı yazılır.
- Çoğu diğer programlama dillerinin aksine ayrı bir derleyici ihtiyacı duymaz. (Ör:C,C++,Java)

- Hem daha okunaklı, hem daha temiz kodsal söz dizimine sahiptir.
- Herhangi bir ücret talebinin olmaması.
- Makine Öğrenmesi için sayıca çok çeşitte paket[1] sahibi olması.
  (Örn :sklearn,statsmodel,...)
- Verinin ön işlemesi (Preproccesing) için çok sayıda kolaylık yaratan paketlerin bulunması.(Örn :pandas,numpy,...)
- SQLite veritabanı ile verimli çalışması.
- Arayüz tasarımında PyQt5 gibi kolay ve anlaşılır bir paket bulunması.

gibi sebeplerden ötürü projemizde Python Programlama Dilini kullandık.

Projemiz için Python'da kullandığımız modüller[2] ve paketler aşağıda belirtilmiştir :

- pandas (Verinin ön işlemesi için)
- numpy (Verinin ön işlemesi için)
- sklearn (Makine Öğrenmesi algoritmaları için)
- sqlite3 (SQLite veritabanını ile python kodunun bağlantısı için)
- PyQt5 (Arayüz tasarımı için)
- sys (PyQt5.QtWidgets.QApplication() fonksiyonunu çalıştırmak ve bazı hata durumları için)
- os (dizin kontrol gibi durumlar için)
- re (regular expressions , yazı dizisindeki ayraçları bulmak için)
- datetime (Zaman aralıkları ve tarih gibi durumlar için)
- passlib.hash (SHA256 algoritması için)
- Ve kodların kullanışlılığı için yaptığımız modüller. Kendi implementasyonumuz olan modüllerin bazıları ileri ki bölümlerde açıklanacaktır.

1 Paket: Python'da bir dizin yapısı içinde bir araya getirilen, birbiriyle bağlantılı modüllere paket adı verilir. Dizin şeklindedir.

2 Modül:Bazı işlevleri kolaylıkla yerine getirmemizi sağlayan birtakım fonksiyonları ve nitelikleri içinde barındıran Python dosyası.

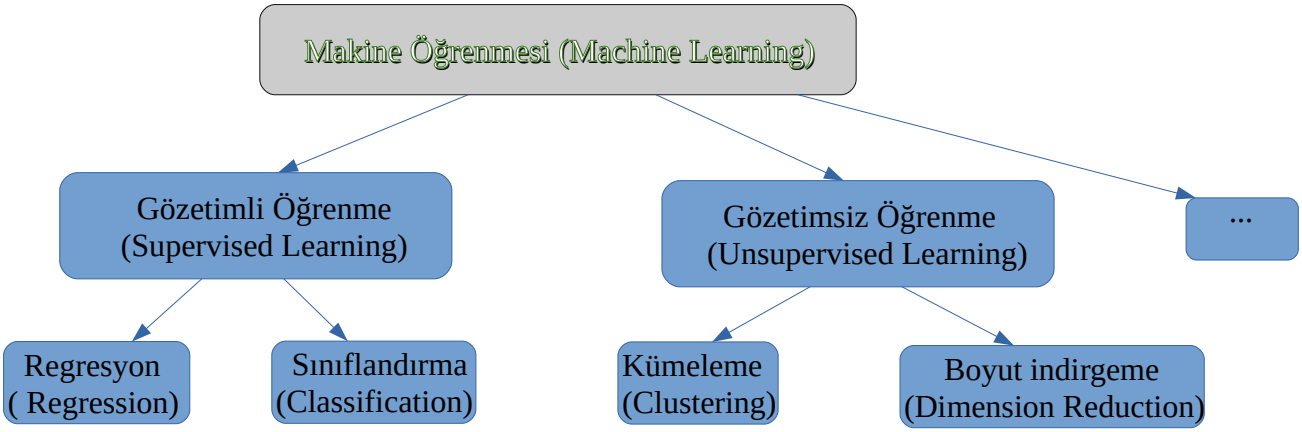
1.3 SQLite Veritabanı

SQLite kullanımı ve kurulumu oldukça basit olan bir veritabanı kütüphanesidir. SQLite veritabanını seçme sebeplerimiz :

- SQLite'ın çalışmak için herhangi bir sunucuya ihtiyacı olmadığından, kurulum veya konfigürasyon adımları yoktur.
- Her veritabanı için sadece bir dosya vardır. Bu da veritabanının yedeklenmesini ve kopyalanmasını kolaylaştırır.
- Platform bağımsızdır.
- Herhangi bir ücret talebinin olmaması.
- Python ile veritabanı arasındaki bağlantılantının kolay bir şekilde kurulabiliyor olması.
- (Kararlı Çapraz Platform Veritabanı)[3] desteğinin sağlanması .
- Ortalama bir programcı için erişilebilir ve okunabilir kaynak kodun bulunması.

1.4 Makine Öğrenmesi (Machine Learning)

Makine Öğrenimi , veritabaları veya veri kümeleri gibi veri türlerine dayalı öğrenimi olanaklı kılan algoritmaların , matematiksel ve istatistiksel teknikler kullanarak verilerden çıkarımlar yapan , tasarım ve geliştirme süreçlerini konu edinen bir bilim dalıdır.



Makine öğrenmesi Gözetimli Öğrenme (Supervised Learning) ve Gözetimsiz Öğrenme (Unsupervised Learning) olmak üzere ikiye ayrılır.

Gözetimli öğrenme algoritmalarında bir bilinmeyen bulunur. Ve mecut bilgiler ile bir metod geliştirilip bu bilinmeyen tahmin edilir. Gözetimsiz Öğrenme algoritmalarında bilinmeyen yoktur. Örneğin bir veri kümesinde belirli bilgiler mevcut ise ve bu bilgilerin

3 Kararlı Çapraz Platform Veritabanı : Herhangi bir makine de yazılan veritabanının ,farklı bir mimariye sahip başka bir makineye kopyalanabilir ve kullanılabilir olmasıdır.

benzer özelliklerine göre gruplanması,kümelenmesi isteniyor ise bu gibi durumlarda Kümeleme algoritmaları kullanılır .

Makine Öğrenimi Gözetimli Öğrenme ve Gözetimsiz Öğrenme dışında başka öğrenme yöntemlerinide barındırır(Ör: Pekiştirmeli Öğrenme (Reincforment Learning)). Yalnız projemizi ilgilendirmediğinden ötürü bahsedilmemiştir. Projemizde sadece Gözetimli Öğrenme algoritmaları kullanılmıştır.

Regresyon (Regression) : Eğer tahmin edilecek değişken bir numerik[4] veri ise , bilinmeyeni tahmin edecek algoritmaya regresyon denilir.

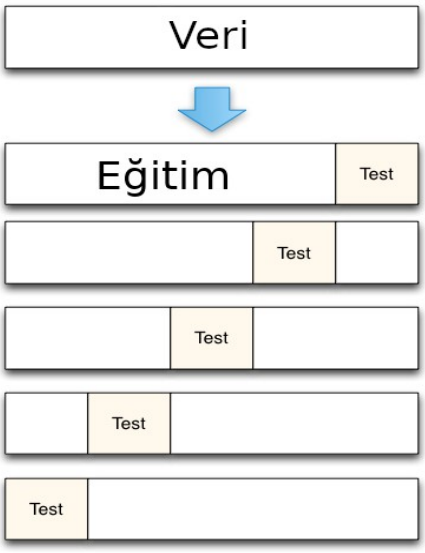
Sınıflandırma (Classification) : Eğer tahmin edilecek değişken bir nominal[5] veri ise , bilinmeyeni tahmin edecek algoritmaya klasifikasyon denilir.

1.5) Algoritmalar

Bu bölümde projemizde kullanmak için seçtiğimiz Makine Öğrenmesi algoritmaları ve bu algoritmalarda en yüksek verim alınabilmesi için kullanılan model seçim algoritması açıklanmıştır.

1.5.1) Model Seçim (Model Selection) Algoritması

Mevcut verinin seçilen algoritmadaki başarısını ölçmek için verinin bir kısmı eğitim için ayrılırken , verinin kalan kısmı test için ayrılır. Buna *Holdout Çapraz Doğrulama(Cross-Validation)* denilir. Genel olarak veri kümesinin $\frac{2}{3}$ 'ü eğitim kümesi $\frac{1}{3}$'ü test kümesi olarak ayrılır. Ve Makine Öğrenmesi algoritmasına uydurulur(fitting).



Görsel 1.1 *K-Katlamalı Çapraz Doğrulma(K-Fold Cross Validation)*

4 Numerik Veri: Sayısal veri anlamına gelir . Sayılar ile ifade edilirler ve "daha fazla" ifadesi ile kullanılabilirler. Nicel veri de denilebilir.Veri kümelerinde genellikle *integer*(tam sayı) ya da *float*(ondalıklı sayı) veri tipinde bulunurlar.

5 Nominal Veri: Kategorik veri anlamına gelir . "daha fazla " ifadesi ile kullanılamazlar. Niteliksel veri de denilebilir. Veri kümelerinde genellikle *string*(yazı dizisi) veri tipinde bulunurlar.

K-Katlamalı Çapraz Doğrulama, *Holdout* yöntemini geliştirmenin bir yoludur. Veri kümesi k alt küme altına bölünmüştür ve bekletme yöntemi k kez tekrarlanmaktadır. Her defasında, k alt kümelerinden biri test kümesi olarak kullanılırken diğer $k^{'}$ alt kümeleri bir eğitim kümesi oluşturmak üzere bir araya getirilir. Ardından, tüm k denemelerindeki ortalama hatası hesaplanır. Bu yöntemin avantajı, verilerin nasıl bölündüğünü daha az önemsemektir.

Oluşturulan modelin başarısı aynı zamanda veri uydurulmadan önce belirtilen bazı parametrelere,özelliklere(features) bağlıdır. En uygun parametreleri bulabilmek için kaba-kuvvete(Brute Force) dayalı *Izgara Araması(GridSearch)* algoritması kullanılır.Bu algoritma denenmesi için belirtilen bütün parametreleri model üzerinde dener ve belirtilen puanlama(Scoring) algoritmasına göre en yüksek puanı alan parametreleri döndürür.

Bahsedilen *K-Katlamalı Çapraz Doğrulama* ile *Izgara Araması* algoritmaları birleştirilirse tahmin edileceği üzere en iyi model bulunacaktır. Bu iki algoritmanın birleşimi olan algoritmaya *GridSearchCV* denilir.Projemizde kullandığımız model seçim algoritması *GridSearchCV'*dir.

1.5.2) Regresyon Algoritmaları

Literatürde sayıca çok Makine Öğrenmesi algoritması bulmak mümkündür. Ama fazla seçim şansının olmasıda hangi algoritmayı hangi durumda kullanmalıyız gibi bir soru ortaya çıkarır. Bu soruya çözüm olarak veri kümesine şu soru sorulur : "Veri doğrusal mı ?" Eğer veri doğrusal ise doğrusal algoritmalar değil ise doğrusal olmayan algoritmalar kullanılır.

Projemizde kullanacağımız regresyon algoritmalarını seçerken üç ayrı algoritma seçmeye karar verdik.Bu algoritmalardan birisi doğrusal problemleri çözmeye odaklı iken diğer iki algoritma doğrusal olmayan problemleri çözmeye odaklıdır. Kullandığımız regresyon algoritmaları ve açıklamaları :

a) Linear Regression(Doğrusal Regresyon)

Kısaca açıklayacak olursak Doğrusal regresyon , lineer olan verilerde kullanılan bir algoritmadır. Bu algoritma veri kümesindeki bütün verilere en ideal uzaklıktaki doğruyu bulabilmeyi amaçlar.

Aşağıda belirtilen denklemlerde Y tahmin edilecek değişkenin kümesi olurken. X bilinenlerin bulunduğu küme olur. N kümedeki eleman sayısını belirtir. $\bar{X}, \bar{Y}$ ise belirtilen kümenin aritmetik ortalaması anlamına gelir. Var , Cov ise varyans ile kovaryans anlamına gelir.

$$Var_X = \frac{1}{N} . \sum_{i=1}^{N} (X_i - \bar{X})^2 \qquad \text{(Denklem 1)}$$

$$Cov_{X,Y} = \frac{1}{N} . \sum_{i=1}^{N} ((X_i - \bar{X}) . (Y_i - \bar{Y})) \qquad \text{(Denklem 2)}$$

$$a = Cov_{X,Y}/Var_X \quad \text{(Denklem 3)}$$

$$b = \bar{Y} - a.\bar{X} \quad \text{(Denklem 4)}$$

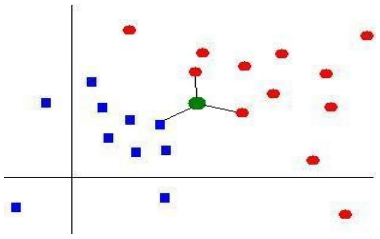
$$y = ax + b \quad \text{(Denklem 5)}$$

Denklem 5'te elde edilen denklem elde edilen Linear Regression doğrunun denklemidir.Bu doğru üzerinden doğrusal problemler için tahminler yapılır.

b) K-Nearest Neighbors Regression(K-En Yakın Komşu Regresyonu)

Algoritmaya göre sınıflandırma sırasında çıkarılan özelliklerden (feature extraction), sınıflandırılmak istenen yeni bireyin daha önceki bireylerden k tanesine yakınlığına bakılmasıdır.Bu yakınlık kavramı şöyle açıklanabilir. Veri kümesinde her satır n-boyutlu kordinat düzleminde bir noktayı temsil eder. Bundan ötürü noktalar arasında uzaklıklar mevcuttur.Burada yeni gelen tahmin verisinin kendine en yakın n komşusunun tahmin değerini verecektir. N-boyutlu olduğundan ötürü noktalar arası uzaklığı bulmak adına çeşitli metrikler bulunur. Projemizde kullandığımız metrikler:

- *Öklit metriği*
- *Manhattan metriği*
- *Chebyshev metriği*
- *Minkowski metriği*

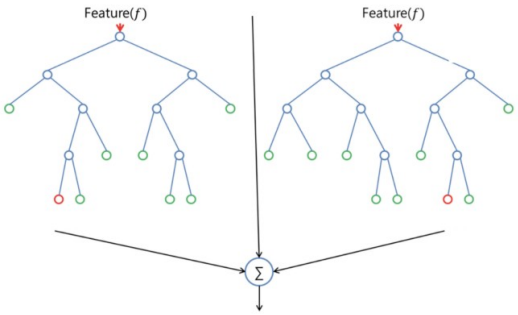


Görsel 1.2 KNN

Algoritmanın bir diğer parametresi ise komşu sayısıdır. En yakın kaç komşuyla ilgilenileceğini belirtir.

c) Random Forest Regression (Rastgele Orman Regresyonu)

Bu algoritma Karar Ağacı(Decision Tree) algoritmasının bir gelişmiş versiyonudur denilebilir. Kısaca açıklanmak istenirse Rastgele Orman, birden fazla karar ağacını oluşturur ve daha doğru ve istikrarlı bir tahmin elde etmek için onları birleştirir.



Görsel 1.3 Random Forest Regression

Karar Ağacını açıklamak istersek , Karar Ağacı veriyi özelliklerine göre parçalara ayırır ve buna göre tahmin yapar .Boy uzunluğu ve kilonun olduğu bir veri kümesinde örneğin 50 kilonun altında olanları bir dal büyük olanları bir dal olacak şekilde kökü ikiye ayırır.Daha sonra ise 150 cm'den uzunları bir dal kısaları ise bir dal olarak uygun alt-daldan ayırır.Ve bu böyle uzar gider. Bu belirtilen bir örnektir , örneklenen şekilde dallanacak diye bir şart yoktur. Rastgele Orman algoritmasını seçmemizin sebebi, Karar Ağacının bizim verimiz gibi az olan verilerde (15 veriden oluşan bir veri kümesi) aşırı uyum(over-fitting) problemi yaratması ve Rastgele Ormanın daha gelişmiş olmasıdır.

1.5.3) Sınıflandırma Algoritmaları

a) K-Nearest Neighbors Classification(K-En Yakın Komşu Sınıflandırması)

1.4.2.b 'de açıklanan algoritmanın aynısıdır. Aralarındaki tek fark bunun nominal veri tahmini yapmasıdır.Bunun sebebide zate sınıflandırma algoritması olmasıdır.

b)Gaussian Naive Bayes Classification (Gauss Naif Bayes Sınıflandırması)

Naif Bayes algoritması ,Bayes teoremi ile veriyi olasılıktan yararlanarak tahmin eder. Bayes teoremi kısaca şöyledir :

$P(A|B) = \frac{P(A \bigcap B)}{P(B)}$ koşullu olasılık olmak üzere;

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

Şöyle ki bir veri kümesinde x bilgi kümesinin elemanı ve y tahmin kümesinin bir elemanı olmak üzere ;

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)} \qquad \text{(Denklem 1)}$$

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y) \qquad \text{(Denklem 2)}$$

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \qquad \text{(Denklem 3)}$$

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow \qquad (1) \qquad \text{(Denklem 4)}$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

2)Yöntem

Projenin yapımı 3 temel parçadan oluşur. İlk aşama Makine Öğrenmesi kodlarının hazırlanması , ikinci aşama kullanıcı verilerinin SQLite veritabanından kullanılabilir hale getirilmesidir. Üçüncü aşama ise PyQt5 ile kullanıcı arayüzünün tasarlanmasıdır.

2.1 Projenin Yapım Basamakları

a) Makine Öğrenmesi Kısmı

a.1. Sahte veri için kod yazılması

a.2. Gelen raw(çiğ) verinin kullanılabilirliğini test eden ve eğer başarılı olursa regresyona uyumlu hale getircek bir fonksiyonun yazılması.

a.3. **a.2**'de işlenmiş veriyi sınıflandırma için kullanılabilir hale getirecek bir fonksiyonun yazılması

a.4. Her aktivite için en iyi regresyon modelini bulacak fonksiyonun yazılması

a.5. En iyi sınıflandırma modelini bulacak fonksiyonun yazılması

a.6. Kullanıcının vereceği bazı bilgiler ile genel olarak hangi aktiviteyi yaptığını tahmin edecek fonksiyonun yazılması.

a.7. Kullanıcının vereceği bazı bilgiler ile hangi aktiviteden yaklaşık yüzde kaç verim alacağını hesaplayan fonksiyonun yazılması.

a.8. Yazılan fonksiyonların birleştirilip bir sınıf objesi haline getirilmesi ve modül halinde kullanılması

b)Veritabanı Kullanırlığı Kısmı

b.1. SQLite kullanımını kolaylaştırmak için yeni bir modül yazılması.

b.2. Veritabanından elde edilecek verinin, tahmin algoritmaları için istenen veri tipine dönüştüren bir modülün kodlanması

c)Kullanıcı Arayüz Tasarım Kısmı

c.1. Kullanıcı için arayüz tasarımının yapılması.

c.2. Kullanıcı girdilerinin veritabanına gönderilmesi.

c.3. Makine Öğrenimi kısmı ile arayüzün birleştirilmesi.

c.4. Kullanıcı giriş ve kayıt formunun tasarlanması.

c.5 Kullanıcı kayıt bilgilerinin saklanacağı veritabanı ile Kayıt formunun bağlantılandırılması.

c.6 Kullanıcı kayıt bilgilerinin saklanacağı veritabanı ile Giriş formunun bağlantılandırılması.

3) Bulgular ve Gerçekleşme

3.1 Makine Öğrenmesi Kısmının Hazırlanması

a.1. Sahte veri için kod yazılması

Projemizde teorik olarak kuracağımız sistemlerde sistemin doğru çalışıp çalışmadığını kontrol etmek amacıyla bir veri oluşturulmuştur. Bu veri ile Makine Öğrenmesinin başarısını ölçmek hedeflenmemiştir. Bu tip başarı ölçeği ancak kullanıcı geri-bildirimleri ile ya da daha büyük veri kümeleri ile olacaktır . Test amaçlı olduğundan bu bölüm ötürü kodlar arasında gösterilmemiştir.

a.2. Gelen raw(çiğ) verinin kullanılabilirliğini test eden ve eğer kullanılabilir olursa regresyona uyumlu hale getircek bir fonksiyonun yazılması

Bu aşamada gelen çiğ veriyi kontrol eden ve eğer mümkünse regresyon için uyumlu hale getiren fonksiyon hazırlanmıştır. Eğer çiğ veri testi geçebilmiş ise fonksiyon aktivite adları ile çiğ verideki aktivite kümelerinin Dataframe[6] halini sözlük veri tipinde döndürür.

a.3. İşlenmiş veriyi sınıflandırma için kullanılabilir hale getirecek bir fonksiyonun yazılması

Veri eğer regresyon için kullanılabilir bir halde ise sınıflandırma için regresyon verisini kullanılabilir bir hale getiren fonksiyon yazılmıştır.Bu fonksiyon sadece bir Dataframe döndürür.Bütün aktivite kümelerinin birleştirilmiş halini.Ama burada verim kolonu silinmiştir.Çünkü tahmin edilecek şey aktivitedir. Bu fonksiyon sadece bir Dataframe döndürür.

a.4. Her aktivite için en iyi regresyon modelini bulacak fonksiyonun yazılması

Bu aşamada **a.2**'de yaptığımız regresyon verisini kullanarak kaba-kuvvet(Brute-Force) tabanlı bir fonksiyon yazıldı. Bu fonksiyon regresyon kümelerinde bulunan her bir aktivite için ayrı olarak, **1.4.1**'de belirtilen regresyon algoritmalarının her birisini hesaplanıp ,en yüksek başarıya sahip olan algoritmayı bulunulan aktivitenin algoritması olarak belirtilir. Ve fonksiyon aktivite isimlerini ve algoritma objelerini sözlük veri tipinde döndürür.

---

6 Dataframe:Veri çerçevesi.Veriyi daha kullanışlı bir hale getiren ,Python'da Pandas paketinde bulunan bir fonksiyon ile oluşturulan veri tipi.

a.5. En iyi sınıflandırma modelini bulacak fonksiyonun yazılması

Bu aşamada **a.3**'te yaptığımız sınıflandırma verisini kullanarak kaba-kuvvet tabanlı bir fonksiyon yazıldı. Bu fonksion **1.4.3**'te belirtilen sınıflandırma algoritmalarının her birisini hesaplayıp en yüksek başarıya sahip olan algoritma sınıflandırma verisinin algoritması olacaktır. Bu fonksiyon sadece algoritma objesini döndürür.

a.6. Kullanıcının vereceği bazı bilgiler ile verdiği durumlarda genel olarak hangi aktiviteyi yapacağını tahmin edecek fonksiyonun yazılması

Bu bölümde **a.5**'te bulunulan algoritmayı kullanan bir fonksiyon yazılmıştır. Bu aşamada kullanıcıya bazı sorular sorulur(arayüzde).Kullanıcıdan elde edilen veriler bu fonksiyona parametre olarak gönderilir ,bunun sonucuda bu fonksiyon genel olarak ,sorulan sorulardaki durumlarda, hangi aktiviteyi yaptığını string veri tipinde döndürür.

a.7. Kullanıcının vereceği bazı bilgiler ile hangi aktiviteden yaklaşık yüzde kaç verim alacağını hesaplayan fonksiyonun yazılması

Bu aşamada **a.4**'te bulunulan algoritmaları kullanarak hangi aktiviteden yüzde kaç verim alınacağını hesaplayan bir fonksiyon yazılmıştır. kullanıcıya daha önce sorulan bazı sorular yeniden sorulur. Kullanıcıdan elde edilen yeni veriler bu fonksiyona parametre olarak gönderilir ,bunun sonucuda fonksiyon aktivite ismi ve verim yüzdelerini dizi içindeki demetler şeklinde döndürür. Demet bir veri tipidir.

a.8. Yazılan fonksiyonların birleştirilip bir sınıf objesi haline getirilmesi ve modül halinde kullanılması

Bu aşama **3.1** bölümde anlatılan fonksiyonların bir sınıf objesi haline getirilmesidir. Bu sayede kullanışlılığı artar , modül olarak çağrıldığında daha verimli çalışır ve daha derli toplu bir kod dizimi olur

3.2 Veritabanı Kullanırlığı Kısmının Hazırlanması

b.1. SQLite kullanımını kolaylaştırmak için yeni bir modül yazılması

Yeni tablo ekleme,bilgi girişi yapma,tablo verisi çekme, vb. durumlar tekrar tekrar kullanılacağından ve uzun veritabanı sorguları olduğundan bu durumları fonksiyonlaştırdık ve bir sınıf altında topladık .İmplement ettiğimiz sınıfı modül olarak **3.3**'de açıklanan konularda kullanılmıştır.

b.2. Veritabanından elde edilecek verinin, tahmin algoritmaları için istenen veri tipine dönüştüren bir modülün kodlanması

Veritabanından çekilecek verinin direkt olarak Makine Öğrenimi için hazırlanan sınıfa sokulması mümkün değildir. Veritabanından gelecek veriyi otomatik olarak istenen şekle getiren bir modül hazırlanmıştır.

3.3 Kullanıcı Arayüz Tasarım Kısmının Hazırlanması

c.1. Kullanıcı için arayüz tasarımının yapılması
Kullanıcının tahmin için girmesi gereken bilgileri girilebileceği, günlük çalışma programını yapabileceği bir arayüz tasarlanmıştır.

c.2. Kullanıcı girdilerinin veritabanına gönderilmesi
Kullanıcının **c.1**'de tasarlanan arayüzde girdiği değerlerin veritabanına kaydedilmesi sağlanmıştır.

c.3. Makine Öğrenimi kısmı ile arayüzün birleştirilmesi
c.1'de tasarlanan arayüze , **3.1**'de oluşturulan Makine Öğrenmesi sınıfının bağlanması ve kullanıcıya hedeflenen hizmetin sunulması.Yalnız bu hizmetin sunulması için bir miktar bilgi girişi gerekmektedir .Bilgi olmadan tahmin olamaz. Bu miktar sağlanmadığı sürece kullanıcı tahmin devrelerini kullanamayacaktır. Bu miktar varsayılan olarak en az iki aktivitenin 15 kere tekrarlanmasıdır. Olay öncesi ve sonrası programa gereken bilgi girişinin sağlanması şarttır. Bu miktar (15 tekrar) tabi ki beş de olabilirdi.Yalnız bu tahmini kötü etkileyecektir. Tekrar sayısını düşürmek için halen araştırmalar yapmaktayız.Bulduğumuz çözümler öneriler kısmında açıklanacaktır.

c.4. Kullanıcı giriş ve kayıt formunun tasarlanması

Kullanıcının **c.1** 'de oluşturulmuş arayüze erişmesi için ilk önce giriş yapması gerekmektedir . Eğer kayıtlı bir kullanıcı değil ise kayıt olması mümkündür. Bu aşamada kullanıcıya giriş ve kayıt olabileceği iki ayrı form arayüz tasarlanmıştır.

c.5 Kullanıcı kayıt bilgilerinin saklanacağı veritabanı ile Kayıt formunun bağlantılandırılması

Kullanıcı eğer kayıt olmak isterse **c.4**'te tasarlananan kayıt formunu doldurmalı ve kaydol butonuna basılmalıdır.Butona basıldığında formdaki bilgiler kontrol edilir ve kullanıcı kayıt bilgilerinin olduğu bir veritabanına gönderilir.Şifre güvenlik amacı ile SHA256 algoritması ile şifrelenip gönderilir.

c.6 Kullanıcı kayıt bilgilerinin saklanacağı veritabanı ile Giriş formunun bağlantılandırılması

Kullanıcı programa giriş yapmak isterse **c.4**'te tasarlanan giriş formu doldurulmalı ve giriş butonuna basılmalıdır. Buton çalıştığı takdirde

doldurulan form kullanıcıların kayıt bilgilerinin olduğu veritabınında kontrol edilir ,eğer bilgiler sağlanırsa giriş yapılır , eğer yanlış bilgi durumu olur ise yanlış bilginin kaynağı belirtilerek kullanıcının tekrar denemesi bekleni

3.4 Proje kodlarının açıklanması

Bu bölümde projemiz için yazdığımız kodların bir kısmını inceledik.Her kısımdan (Makine Öğrenim kısmı,Veritabanı kullanırlığı kısmı ve Arayüz tasarım kısmı) sadece bir tane kod dosyası açıklanmıştır.Bunun sebebi kod sayısının fazla olması bu sebepten ötürü proje raporunda belirtilmesinin imkanının olmamasıdır. Bütün kodlar pdf halinde , dökümanlar kısmında belirtilmiştir.

3.4.1 preproccesingToDB.py(Veritabanı kullanırlığı b.2 kısmının kodları)

preproccesingToDB.py

```python
import pandas as pd
from sqliteModule import sql_functions
import os
class preproccesedData():
    def __init__(self,dayTableColumns,sleepTableColumns,rawDatabasePath, *args, **kwargs):
        self.dayTableColumns = dayTableColumns
        self.sleepTableColumns = sleepTableColumns
        if os.path.isfile(rawDatabasePath):
            self.rawDatabasePath = rawDatabasePath
            self.__RawDB = sql_functions(self.rawDatabasePath)
        else:
            raise FileNotFoundError("Geçersiz veritabanı konumu.")
        self.__data_dict = dict()

    def __databaseToArray(self):
        for table in self.__RawDB.tableList():
            table = table[0]
            data = self.__RawDB.get_whole_table(table)
            for iteration in data:
                raw = list(iteration)
                if table != "SleepTable":
                    job = iteration[0]
                    raw.pop(0)
                    raw.insert(0,table)
                else:
                    job = table
                try:
                    self.__data_dict[job].append(raw)
                except KeyError:
                    self.__data_dict[job] = list()
                    self.__data_dict[job].append(raw)
    def __arrayToDataFrame(self):#istihza adlı kitapta kodların obje kullanıcısına gizlemeyi öğren..
        for key in self.__data_dict:
            info = self.__data_dict[key]
            if key == "SleepTable":
                df = pd.DataFrame(data = info,index = range(len(info)),columns = self.sleepTableColumns)
            else:
                df = pd.DataFrame(data=info,index = range(len(info)),columns = self.dayTableColumns)
            self.__data_dict[key] = df
    @property
    def get_data_Dict(self):
        self.__databaseToArray()
        self.__arrayToDataFrame()
        return self.__data_dict

```

Kolon bilgileri ile veritabanı konumunun kontrolü gerçekleşir.

Not:Eğer bir sınıfta fonksiyonun başın “__” gelirse fonksiyon gizli olur.

Veritabanından gelen bilgileri diziye çevirir.

Diziye çevrilen verileri pandas DataFrame’ine çevirir.

Makine öğrenimi sınıfının (**3.4.1**) istediği veriyi döndürür.

3.4.2 ML_Procces.py(Makine Öğrenimi kısmının kodları)

ML_Process.py

```
##################################
#Preprocessing
import pandas as pd
import numpy as np
##################################
#Regressions
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
##################################
#Clasifications
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
##################################
#Model Selection
from sklearn.model_selection import GridSearchCV
##################################

class ML():
    def __init__(self,data_dict,*args, **kwargs):
        self.data_dict = data_dict
        self.predictable_data_dict = self.findPredictableFrames()
        if self.predictable_data_dict != dict():
            self.generalClassificationDataset = self.generalSituationsDataset

            self.algorithmDict = self.findBestRegression

            self.classificationEstimator = self.findBestClassification

    def findPredictableFrames(self):
        predictable_data_dict = dict()
        for key in self.data_dict:
            if key != "SleepTable":
                if len(self.data_dict[key].index) < 15:
                    pass
                else:
                    if ("Date" and "TimeRange") in list(self.data_dict[key].columns):
                        self.data_dict[key] = self.data_dict[key].drop(["Date","TimeRange"],axis = 1)
                    predictable_data_dict[key] = self.data_dict[key]
        return predictable_data_dict

    @property
    def generalSituationsDataset(self):
        generalDF = pd.DataFrame()
        for key in self.predictable_data_dict:
            df = self.predictable_data_dict[key]
            job = key
            y = [job for i in range(len(df.index))]
            y = pd.DataFrame(y)
            newDf = pd.concat([df,y],axis = 1) # axis = 1 ==> X-axis ; axis = 0 ==> Y-axis
            generalDF = pd.concat([generalDF,newDf],axis = 0)
        generalDF.columns = list(df.columns)+["Job"]
        generalDF.index = range(len(generalDF.index))

        generalDF.drop("Efficiency",axis = 1,inplace = True)
        return generalDF

    @property
    def findBestRegression(self):
        algorithmDict = dict()
        for key in self.predictable_data_dict:
            algorthm = self.bestRegression(self.predictable_data_dict[key])
            algorithmDict[key] = algorthm
        return algorithmDict

    def bestRegression(self,specifiedDataFrame):
        X = specifiedDataFrame.iloc[:,:-1]
        Y = np.ravel(specifiedDataFrame.iloc[:,-1:].values)
        modelResults = list()
        res1 = self.LinRegConclusion(X,Y)
        res2 = self.KNNConclusion(X,Y)
        res3 = self.RandomForestConclusion(X,Y)
        modelResults.append(res1)
        modelResults.append(res2)
        modelResults.append(res3)

        return max(modelResults,key = lambda tup:tup[0])

    def LinRegConclusion(self,X,Y):
        paramsLinearRegression = [{}]
        LinearRegressionSearcher = GridSearchCV(estimator=LinearRegression(),param_grid =paramsLinearRegression,scoring = 'r2',cv = 5)
        LinearRegressionSearcher.fit(X,Y)
        LinearRegressionScore = LinearRegressionSearcher.best_score_
        LinearRegressionestimator = LinearRegressionSearcher.best_estimator_

        return (LinearRegressionScore,LinearRegressionestimator)

    def RandomForestConclusion(self,X,Y):
        paramsRandomForestRegressor = [{"n_estimators":list(range(5,20))}]
        RandomForestRegressorSearcher = GridSearchCV(estimator=RandomForestRegressor(),param_grid =paramsRandomForestRegressor,scoring = 'r2',cv = 5)
        RandomForestRegressorSearcher.fit(X,Y)
        RandomForestRegressorScore = RandomForestRegressorSearcher.best_score_
        RandomForestRegressorestimator = RandomForestRegressorSearcher.best_estimator_

        return (RandomForestRegressorScore,RandomForestRegressorestimator)

    def KNNConclusion(self,X,Y):
        paramsKNeighborsRegressor = [{"metric":["euclidean","minkowski","chebyshev","manhattan"],"n_neighbors":list(range(1,5))}]
        KNeighborsRegressorSearcher = GridSearchCV(estimator=KNeighborsRegressor(),param_grid =paramsKNeighborsRegressor,scoring = 'r2',cv = 5)
        KNeighborsRegressorSearcher.fit(X,Y)
        KNeighborsRegressorScore = KNeighborsRegressorSearcher.best_score_
        KNeighborsRegressorestimator = KNeighborsRegressorSearcher.best_estimator_

        return (KNeighborsRegressorScore,KNeighborsRegressorestimator)
```

Gereken modül ve paketlerin yüklenmesi.

Gelen verinin ,data_dict, boş olup Olmadığını kontrol eder ve fonksiyonları ona göre çalıştırır.

a.2'nin Python kodudur. Ek olarak bazı gereksiz kolonların silimi gerçek-leşir.

a.3'ün Python kodudur. Sınıflandırma için tek bir veri kümesi hazırlar.

FindBestRegression fonksiyonu **a.4**'ün Python kodudur.En iyi regresyon modellerini bulmaya yarar.Tek başına çalışmaz . Görüleceği üzere sınıftaki diğer fonksiyonlara dayalı bir fonksiyondur .


```
    def giveMeAdvice(self,data):
        results = list()
        for key in self.algorithmDict:
            obj = self.algorithmDict[key][1]
            efficiency = obj.predict(data)
            results.append((efficiency,key))
        adviceList = sorted(results,key = lambda tup:tup[0],reverse = True)
        return adviceList

    @property
    def findBestClassification(self):
        X = self.generalClassificationDataset.iloc[:,:-1]
        Y = np.ravel(self.generalClassificationDataset.iloc[:,-1:].values)
        res1 = self.GaussianNBConclusion(X,Y)
        res2 = self.KNNClasficationConclusion(X,Y)
        bestClassification = max([res1,res2],key = lambda tup:tup[0])
        classificationEstimator = bestClassification[1]
        return classificationEstimator

    def GaussianNBConclusion(self,X,Y):
        paramsGaussianNB = [{}]
        GaussianNBSearcher = GridSearchCV(estimator=KNeighborsClassifier(),param_grid =paramsGaussianNB,scoring = 'accuracy',cv = 5)
        GaussianNBSearcher.fit(X,Y)
        GaussianNBScore = GaussianNBSearcher.best_score_
        GaussianNBestimator = GaussianNBSearcher.best_estimator_
        return (GaussianNBScore,GaussianNBestimator)

    def KNNClasficationConclusion(self,X,Y):
        paramsKNeighborsClassification = [{"metric":["euclidean","minkowski","chebyshev","manhattan"],"n_neighbors":list(range(1,5))}]
        KNeighborsClassificationSearcher = GridSearchCV(estimator=KNeighborsClassifier(),param_grid =paramsKNeighborsClassification,scoring = 'accuracy',cv = 5)
        KNeighborsClassificationSearcher.fit(X,Y)
        KNeighborsClassificationScore = KNeighborsClassificationSearcher.best_score_
        KNeighborsClassificationestimator = KNeighborsClassificationSearcher.best_estimator_
        return (KNeighborsClassificationScore,KNeighborsClassificationestimator)

    def GiveGeneralActivity(self,data):
        activity = self.classificationEstimator.predict(data)
        return activity
```

Yukarda açıklanan ML_Procces.py dosyasına ek olarak bir iki şey daha ekleyecek olursak,bazı fonksiyonların üzerinde bulunan *@property* bezeyicisinin(Decorator) anlamı , altındaki fonksiyonu bir özellik (Attribute) haline getirmektir . Ama bu şekilde özellik olan bir özelliğin farkı vardır. Bu özellik salt okunurdur. Kısacası fonksiyon dışında değiştirilmesine izin yoktur. Bir diğer konu ise şunlardır :

- *LinRegConclusion*
- *RandomForestConclusion*
- *KNNConclusion*
- *GaussianNBConclusion*
- *KNNClassificationConclusion*

fonksiyonlarıdır.Bu fonksiyonlar **1.4**'te açıklanan algoritmalardır.Ve son olarak **1.4.1**'deki model seçimi Python'da *GridSearchCV* fonksiyonu ile yapılmıştır.

3.4.3 predictionPage.py(Arayüz tasarım kısmının c.1 kısmı kodlarının bir parçası)

Bu kod parçası tahmin formunun arayüz kodlarıdır. **3.4.1**'deki kod dosyasından yararlanılır . Bu kod dosyasının arayüz görüntüsü **3.4.4**'teki **Görsel 5** 'te görülmektedir.

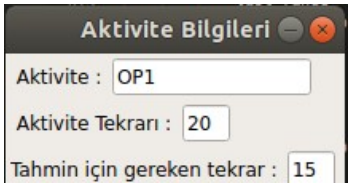
```python
#Arayüz Tasarım modülleri
from PyQt5.QtWidgets import (QWidget,QLineEdit,QLabel,QHBoxLayout,QVBoxLayout,
                             QTableView,QPushButton,QGraphicsDropShadowEffect,QApplication)
from PyQt5.QtGui import QStandardItemModel,QStandardItem
from PyQt5.QtCore import Qt,QTimer
#Makine Öğrenmesi Modülü
from ML_Process import ML
#Sistem modülü
import sys

class PredictionForm(QWidget):
    def __init__(self,data_dict,*args,**kwargs):
        super(PredictionForm, data_dict: dict).__()
        self.data_dict = data_dict #Parametre olan veri kümesinin sınıf attribute'si olması
        self.mlObj = ML(self.data_dict)#Makine öğrenmesi sınıfına parametre olarak verinin gönderilmesi ve durumun
        # hesaplanması.Bu satırın çalışması yaklaşık 7-10 saniye sürer.Çünkü obje çağrıldığında brute-force ile
        # algoritmalar hesaplanır.
        self.setWindowTitle("Tahmin Formu")#Pencere Başlığı
        self.setMaximumSize(386,409)#Pemcere boyutları
        self.initUI()#Arayüz tasarım kodlarının çağrılması

    def initUI(self):
        #Layout'ların tanımlanması
        self.mainVerticalLayout = QVBoxLayout()
        self.mainVerticalLayout.setContentsMargins(6,6,6,6)
        self.headerLayout = QHBoxLayout()
        self.willingnessLayout = QHBoxLayout()
        self.fatigueLayout = QHBoxLayout()
        self.moraleLayout = QHBoxLayout()
        self.adviceLayout = QHBoxLayout()
        self.tableLayout = QHBoxLayout()

        #Widget'lerin tanımlanması,tasarlanması ve belirlenen layout'lara eklenmesi

        self.header = QLabel("Tahmin Formu")
        self.header.setStyleSheet("""background-color:#E2E7EE;color : #660022;
                font-size:32px;font-family: Courier;
                padding : 4px;""")
        self.headerLayout.addStretch()
        self.headerLayout.addWidget(self.header)
        self.headerLayout.addStretch()

        styleSheetLabel = """font-family:Courier;font-weight:650;font-size:14px;
        color:#CDF1F9;background-color:#48234f;
        border-right:4px solid #065535;border-left:4px solid #065535;padding : 2px;"""
        styleSheetLineEdit = """font-family:Consolas;font-weight:550;font-size:12px;
        background-color:#400080;color:#FFF096;
        border:3px solid #96a5ff;padding : 2px;border-radius:3px;"""

        self.willingnessLabel = QLabel("Çalışma isteği")
        self.willingnessLabel.setStyleSheet(styleSheetLabel)
        self.willingnessLineEdit = QLineEdit()
        self.willingnessLineEdit.setStyleSheet(styleSheetLineEdit)
        self.willingnessLineEdit.setMaximumWidth(150)
        self.willingnessLineEdit.setMinimumWidth(150)

        self.willingnessLayout.addStretch()
        self.willingnessLayout.addWidget(self.willingnessLabel)
        self.willingnessLayout.addWidget(self.willingnessLineEdit)

        self.fatigueLabel = QLabel("Hissedilen yorgunluk")
        self.fatigueLabel.setStyleSheet(styleSheetLabel)
        self.fatigueLineEdit = QLineEdit()
        self.fatigueLineEdit.setStyleSheet(styleSheetLineEdit)
        self.fatigueLineEdit.setMaximumWidth(150)
        self.fatigueLineEdit.setMinimumWidth(150)

        self.fatigueLayout.addStretch()
        self.fatigueLayout.addWidget(self.fatigueLineEdit)

        self.moraleLabel = QLabel("Moral")
        self.moraleLabel.setStyleSheet(styleSheetLabel)
        self.moraleLineEdit = QLineEdit()
        self.moraleLineEdit.setStyleSheet(styleSheetLineEdit)
        self.moraleLineEdit.setMaximumWidth(150)
        self.moraleLineEdit.setMinimumWidth(150)

        self.moraleLayout.addStretch()
        self.moraleLayout.addWidget(self.moraleLabel)
        self.moraleLayout.addWidget(self.moraleLineEdit)

        self.adviceButton = QPushButton("Tahminleri göster")
        self.adviceButton.setStyleSheet("""font-family:Courier;font-weight:650;font-size:14px;
        color: #ff903e;background-color:#00ffc1;
        border:1px solid #ffc100;border-radius:3px;padding:2px;""")
        shadow = QGraphicsDropShadowEffect(blurRadius=6, xOffset=5, yOffset=5)
        self.adviceButton.clicked.connect(self.advice)#Tahminleri gösterecek fonksiyonun çağrılması
        self.adviceButton.setGraphicsEffect(shadow)

        self.adviceLayout.addStretch()
        self.adviceLayout.addWidget(self.adviceButton)
        self.adviceLayout.addStretch()

        self.adviceResults1Header = QLabel("Çalışma Verimi Tahminleri")
        self.adviceResults1Header.setStyleSheet("""font-family:Courier;font-size:16px;font-weight:650;
        background-color : #725e51;color:#ffeeed;
        border-left:8px solid #065535;padding : 4px;""")

        self.adviceResults1Header.setMaximumWidth(300)
        self.adviceResults1Header.setHidden(True)

        self.adviceResults2Header = QLabel("Genel Aktivite Tahmini")
        self.adviceResults2Header.setStyleSheet("""font-family:Courier;font-size:16px;font-weight:650;
        background-color :  #516572;color:#ffeeed;
        border-left:8px solid #065535;padding : 4px;""")
        self.adviceResults2Header.setMaximumWidth(300)

        self.adviceResults2Header.setHidden(True)

        self.table_view = QTableView()
        self.table_view.setHidden(True)
        self.tableLayout.addStretch()
        self.tableLayout.addWidget(self.table_view)
        self.tableLayout.addStretch()

        self.generalActLayout = QHBoxLayout()
        self.generalActLabel = QLabel()
        self.generalActLayout.addStretch()
        self.generalActLayout.addWidget(self.generalActLabel)
        self.generalActLayout.addStretch()

        self.mouseDoubleClickEvent = self.mouseDoubleClickEvent#Bu satır olmaz ise interface bu widgeti göstermez

        #Tasarlanan layout'ların ana layout'a eklenmesi
        self.mainVerticalLayout.addStretch()
        self.mainVerticalLayout.addLayout(self.headerLayout)
        self.mainVerticalLayout.addLayout(self.willingnessLayout)
        self.mainVerticalLayout.addLayout(self.fatigueLayout)
        self.mainVerticalLayout.addLayout(self.moraleLayout)
        self.mainVerticalLayout.addLayout(self.adviceLayout)
        self.mainVerticalLayout.addWidget(self.adviceResults1Header)
        self.mainVerticalLayout.addLayout(self.tableLayout)
        self.mainVerticalLayout.addWidget(self.adviceResults2Header)
        self.mainVerticalLayout.addLayout(self.generalActLayout)
        self.mainVerticalLayout.addStretch()
```

```
        self.setLayout(self.mainVerticalLayout)

    def efficiencyPredictionTable(self,data):
        #Bu fonksiyon veriyi tablo modeline çevirir ve modeli table_view'e atamasını yapar.
        model = QStandardItemModel()
        model.setRowCount(len(data))
        model.setColumnCount(1)
        model.setHorizontalHeaderLabels(["Verim"])
        rows = [str(d[0]) for d in data]
        model.setVerticalHeaderLabels(rows)

        index = 0
        for job,efficiency in data:
            item = QStandardItem("%"+str(efficiency))
            item.setFlags(Qt.ItemIsEnabled)
            model.setItem(index,0,item)
            index += 1

        self.table_view.setModel(model)
        self.table_view.setColumnWidth(0,40)
        self.table_view.setMaximumWidth(73)
        self.table_view.setMaximumHeight(111)
        self.table_view.setHidden(False)

    def advice(self):
        willingness = self.willingnessLineEdit.text()
        fatigue = self.fatigueLineEdit.text()
        morale = self.moraleLineEdit.text()
        #Hataların handle edilmesi
        try:
            willingness = int(willingness)
            if not(isinstance(willingness,int)):
                self.willingnessLineEdit.setText("1 ile 100 arasında bir sayı giriniz")
                return ""

            fatigue = int(fatigue)
            if not(isinstance(fatigue,int)):
                self.fatigueLineEdit.setText("1 ile 100 arasında bir sayı giriniz")
                return ""

            morale = int(morale)
            if not(isinstance(morale,int)):
                self.moraleLineEdit.setText("1 ile 100 arasında bir sayı giriniz")
                return ""

        except ValueError:
            return ""

        data =[[willingness,fatigue,morale]]
        adviceList = self.mlObj.giveMeAdvice(data)#Aktiviteye dayalı verimlerin bilgisini elde eder
        generalActivity = self.mlObj.GiveGeneralActivity(data)[0]#Genel aktivite ismini elde eder
        adviceList = [(job,round(float(efficiency))) for efficiency , job in adviceList]
        self.adviceResults1Header.setHidden(False)
        self.adviceResults2Header.setHidden(False)
        self.efficiencyPredictionTable(adviceList)
        self.generalActLabel.setStyleSheet("""
        font-family:Consolas;font-size:14px;font-weight:650;
        color:#e5e5e5;background-color:#555555;
        border:1px solid #065535;padding:2px;
        """)
        self.generalActLabel.setText(generalActivity)
```

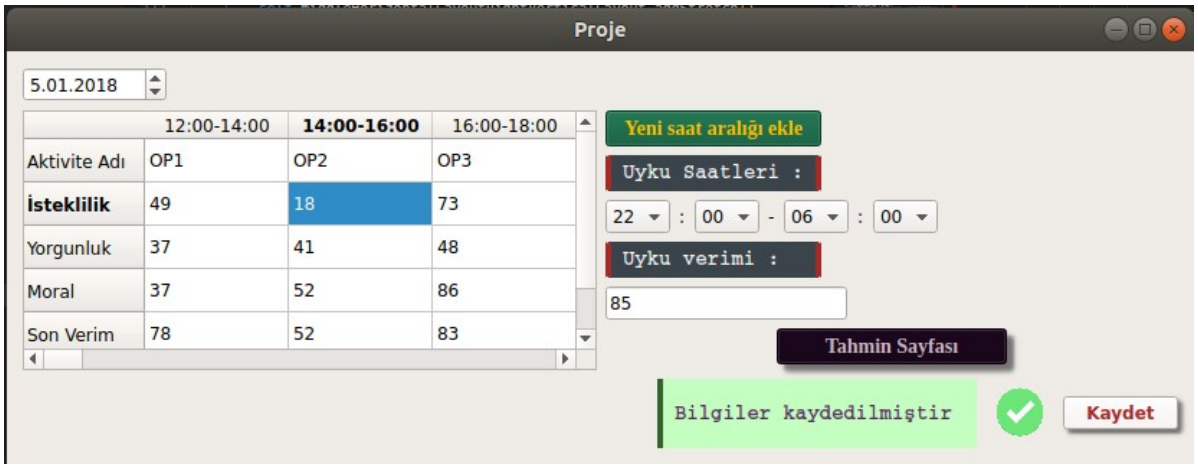
3.5 Proje Ekran Görüntüleri

Görsel 1. Kayıt Formu

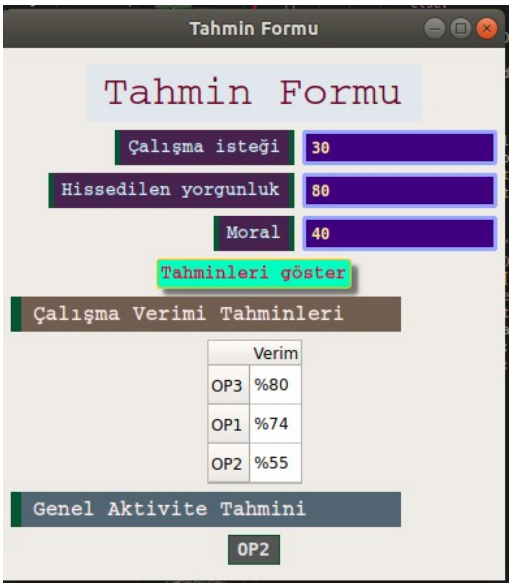
Görsel 2. Kullanıcı Girişi



Görsel 3. Aktivite Bilgisi



Görsel 4.Ana Ekran



Görsel 5.Tahmin Ekranı

4. Sonuçlar ve Tartışma

Projemiz kullanıcıya günlük çalışma programını hazırlayabileceği bir arayüz, bu arayüze girilen bilgiler aracılığı ile aktivite verimleri ve genel aktivite tahminlerini sunar. Yaptığımız program sayesinde kullanıcı , kendisine daha uygun çalışma programları hazırlayabilmektedir.

Projeyi yaparken bazı hatalar ile karşılaştık. Bunlardan birisi değişkenler arasına zorluk değişkenini eklememiz oldu.Bu değişken gibi aktiviteye bağlı değişkenler ,hep aynı değeri alırlar, bundan ötürü algoritmanın tahminine zarar verirler. Bu sebepten zorluk gibi aktiviteye bağlı değişkenleri, değişken listemize eklemedik. Ama bu demek değildir ki bu değişkenler kullanılamaz . Halen bu konu üzerinde araştırmalar yapmaktayız.

Karşılaştığımız bir diğer problem ise değişkenlere saat aralığını eklediğimiz zaman oldu. Saat aralıkları işlenemez sayıda olduğundan ,ilk önce günü parçalara ayırıp ayırılan parçalardan hangilerine dahil olur şeklinde bir deneme yaptık. Bunun sonucunda yine

başarısızlıkla karşılaştık çünkü günün parçalara ayrılışı kadar veri kümesine kolon eklenir ve bu kolonların değerleri ya 1 ya da 0 olur. Ve bazen aynı satırda birden fazla 1 olabilir.Bu gibi durumlar Makine Öğrenmesi algoritmalarımızda problem yarattığından saat aralığı değişkenide çıkarılan değişkenler listemize girdi.

Yaptığımız masaüstü uygulaması kullanıcılar tarafından beğenilmiştir. Ama aktivite tekrar sayısının fazlalığı kullanımı zorlaştırmaktadır. Aktivite saysını azaltmak için halen araştırmalar yapmaktayız.

5.Öneriler

Projenin bir sonraki aşamsında masaüstü uygulaması değil de Android ya da Web uygulaması yapılabilir.Bu sayede daha çok kitleye hitap etmesi mümkündür.

Projenin yüksek kullanıcıya hitap etmesi durumunda aktivite tekrar sayısı düşürülebilir. Şöyle ki bir kullanıcı için yapılacak tahmin için yaratılacak veri kümesinin bir miktarını diğer kullanıcıların veri kümesinden yararlanarak geri kalanını da kullanıcıdan alarak yapılır ve bu sayede aktivite tekrar sayısı azaltılır. Yalnız bu durumlar sınırdışı[7](outlayer) olan kullanıcıların verim tahminlerine zarar verecektir .

Projenin yüksek kullanıcıya hitap etmesi durumunda elde edilen veriler ,veri sahibinin gizliliği ve kullanıcı izni dahilinde, psikolojik,sosyolojik vb. araştırmalar için açık kaynak olarak yayınlanabilir.

Projede Derin Öğrenme'nin kullanılması ile verim tahminini geliştirmek mümkündür.

Kaynakça

1. Raschka Sebastian,Python Machine Learning,2015,Packt Publishing (9 Kasım 2018 Tarihinde erişildi)
2. Başer Mustafa ,Python,2018,Dikeyeksen Yayıncılık (9 Kasım 2018 Tarihinde erişildi)
3. https://belgeler.yazbel.com/python-istihza/ (10 Kasım 2018 Tarihinde erişildi)
4. https://en.wikipedia.org/wiki/Conditional_probability (10 Kasım 2018 Tarihinde erişildi)
5. https://en.wikipedia.org/wiki/Bayes%27_theorem (10 Kasım 2018 Tarihinde erişildi)
6. https://scikit-learn.org/stable/modules/naive_bayes.html#gaussian-naive-bayes (10 Kasım 2018 Tarihinde erişildi)
7. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (11 Kasım 2018 Tarihinde erişildi)
8. https://scikit-learn.org/stable/modules/neighbors.html (12 Kasım 2018 Tarihinde erişildi)
9. http://madhugnadig.com/articles/machine-learning/2017/03/04/implementing-k-meansclustering-from-scratch-in-python.html (12 Kasım 2018 Tarihinde erişildi)

7 Sınırdışı(Outlayer):Standartları normalin dışında olan veriye denir.

10. https://machinelearningmastery.com/implement-simple-linear-regression-scratch-python/ (13 Kasım 2018 Tarihinde erişildi)
11. https://www.sqlite.org/different.html (14 Kasım 2018 Tarihinde erişildi)
12. https://www.cyberciti.biz/python-tutorials/securely-hash-passwords-in-python/ (14 Kasım 2018 Tarihinde erişildi)
13. https://likegeeks.com/pyqt5-tutorial/ (15 Kasım Tarihinde erişildi)
14. https://stackoverflow.com/questions/685206/how-to-get-a-list-of-column-names (4 Aralık 2018 Tarihinde erişildi)
15. https://stackoverflow.com/questions/7727863/how-to-make-a-cell-in-a-qtablewidget-readonly (4 Aralık 2018 Tarihinde erişildi)
16. https://stackoverflow.com/questions/51619186/pyqt5-qtablewidget-right-click-on-cell-wontspawn-qmenu (5 Aralık 2018 Tarihinde erişildi)
17. https://stackoverflow.com/questions/44264157/pyqt-add-right-click-to-a-widget (8 Aralık 2018 Tarihinde erişildi)
18. https://github.com/RavenKyu/OpenTutorials_PyQt/blob/master/QtFramework/QtWidgets/QComboBox/QComboBox_03_model_tableview.py (12 Aralık 2018 Tarihinde erişildi)