

**İSTANBUL TEKNİK ÜNİVERSİTESİ**  
**Bilgisayar ve Bilişim Fakültesi**

**Makine Öğrenmesi: Film Öneri Sistemi**

**STAJ**  
**Kadir Emre Oto**  
**150140032**

**YAZ / 2017**

**İstanbul Teknik Üniversitesi**  
**Bilgisayar ve Bilişim Fakültesi**  
**STAJ RAPORU**

Akademik Yıl: 2017

Staj yapılan dönem: ☒Yaz ☐Bahar ☐Güz

**Öğrenci ile ilgili bilgiler**

Adı ve Soyadı: Kadir Emre Oto

Öğrenci Numarası: 150140032

Bölüm: Bilgisayar Mühendisliği

Program: %30 İngilizce

E-posta Adresi: [otok@itu.edu.tr](mailto:otok@itu.edu.tr)

(Cep) Tel No: 0 (507) 713 83 90

ÇAP öğrencisi misiniz? ☐ Evet (ÇAP yaptığınız Fakülte/Bölüm: \_\_\_\_\_)

☒ Hayır

Mezuniyet  
durumunda mısınız? ☐ Evet

☒ Hayır

Yaz okulunda ders ☐ Evet (Ders sayısı: \_\_)

alıyor musunuz?

☒ Hayır

**Öğrencinin çalıştığı kurum ile ilgili bilgiler**

İsmi: Erstream Video Delivery Company

Birimi: Yazılım Departmanı

Web Adresi: <http://erstream.com>

Kısa Adresi: Defterdar Mah. Otakçılar Cad. 80/9, Eyup | 34050 İstanbul | Turkey

**Yetkili kiři ile ilgili bilgiler**

Bölümü: Yazılım Departmanı  
Unvanı: Bilgisayar Mühendisi  
Adı ve Soyadı: Beysim Gayret  
(Kurumsal) E-posta: [beysim.gayret@erstream.com](mailto:beysim.gayret@erstream.com)  
(Kurumsal) Tel. No.: +90 212 705 95 17

**Yapılan iş ile ilgili bilgiler**

Staj yeri ☒Türkiye  
☐Yurtdışı  
Staj başlangıç tarihi 19.06.2017  
Staj bitiş tarihi 21.07.2017  
Stajda çalışılan net **gün** sayısı 20  
Staj süresince sigortanız var mıydı? ☒Evet, İTÜ tarafından sigortalandım.  
☐Evet, kurum tarafından sigortalandım.  
☐Hayır, yurtdışı stajı yaptım.  
☐Hayır.

## İÇİNDEKİLER

1. KURULUŞ HAKKINDA BİLGİLER.....	1
2. GİRİŞ.....	2
3. STAJ PROJESİNİN TANIMI VE ANALİZİ.....	3
3.1 Araştırma.....	3
3.2 Geliştirme.....	6
4. SONUÇ.....	9
5. REFERANSLAR.....	10
6. EKLER.....	11

## **KURULUŞ HAKKINDA BİLGİLER**

Uğur Kalaba, Radoslav Raychev ve Yağız Buran tarafından 2003 tarihinde kurulmuş olan Erstream Video Delivery Company internet video teknolojileri alanında ürettiği ürünler ve sunduğu hizmetler ile bu alanda öncü firmalardan biri olmuştur. Dünya çapında ün yapmış içerik sahipleri ve yayıncılar ile birlikte Türkiye’de yayıncılık yapan bir çok firma Erstream ile işbirliği içerisinde çalışmaktadır.

Erstream’in dünya çapında yüzlerce internet ve televizyon kanalına yayın sağlayabilmesinin temel nedeni Avrupa’daki bir çok ülkenin veri merkezlerinde encoding ve downlink hizmetlerine sahip olmasıdır.



**Şekil 1 – Erstream Logosu**

## **GİRİŞ**

Ersteam staj programına kabul ettiđi stajyerlere řirketin üzerinde alıřtıđı bir ok rn ve sađladıđı hizmetler üzerinde alıřma imkanı sađlamakta ve tecrbelerini aktarmayı hedeflemektedir. Bunlara ek olarak eřitli arařtırma-geliřtirme (AR-GE) projelerinde de alıřma imkanı sunmaktadır.

Stajımda bir AR-GE projesi olarak makine ğrenmesi ile kullanıcılara izledikleri filmlere gre yeni filmler neren bir sistem geliřtirdim. Proje üzerinde alıřan bařka kimse olmadığı iin makine ğrenmesi ve film neri sistemleri üzerinde detaylı arařtırmalar ve denemeler yapılması gerekiyordu. Bylelikle uzmanı olmadığı bir alanda nasıl proje geliřtirebileceđim hakkında tecrbe kazanma fırsatı elde ettim.

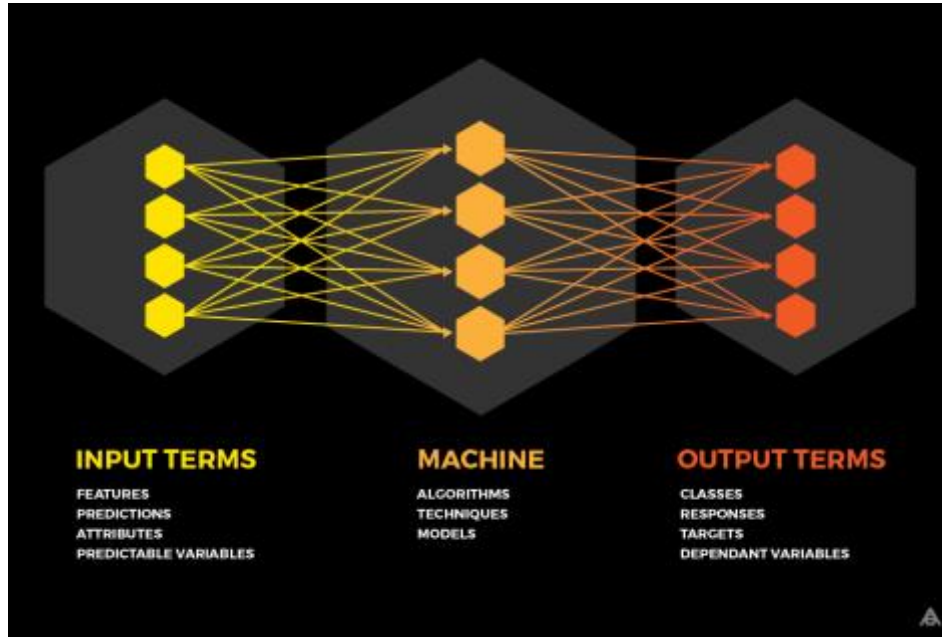
## STAJ PROJESİNİN TANIMI VE ANALİZİ

Stajda, kullacıların filmlere verdikleri puanlar, filmi ne kadar izledikleri gibi verileri kullanarak makine öğrenmesi ile yeni filmler öneren bir araştırma ve geliştirme (**AR-GE**) projesi yapmam bekleniyordur. Stajımı bu doğrultuda araştırma ve geliştirme süreçleri olmak üzere iki kısma ayırmak mümkündür.

### 1.1 Araştırma

Projenin ilk aşaması makine öğrenmesinin araştırılması ve projeye uygun kütüphanenin belirlenmesinden oluşuyor. Araştırma aşamasında çeşitli internet

**Makine öğrenmesi**, bilgisayarların açık bir şekilde programlanmadan önceden elde edilmiş verileri kullanarak çıkarımlar yapmasını sağlayan bir bilim tekniğidir. Son 10 yılda popülerliği oldukça artmış ve bir çok alanda kullanılmaya başlanmıştır. Örneğin gereksiz veya saldırı amacı taşıyan e-postaların belirlenmesi ve kullanıcıların uyarılmasında makine öğrenmesi kullanılarak yazılmış filtreler kullanılır ve kullacıların güvenliklerinin artırılması amaçlanır. Çeşitli vücut verilerinin incelenip insanların kansere yakalanma olasılığın belirlenmesi gibi sağlık alanlarında makine öğrenmesi kullanılabilir. Film sektöründe önceden izlenen filmleri kullanarak izleyicilere beğenebilecekleri yeni filmlerin önerilmesinde makine öğrenmesi algoritmaları da kullanılmaya başlanmıştır.

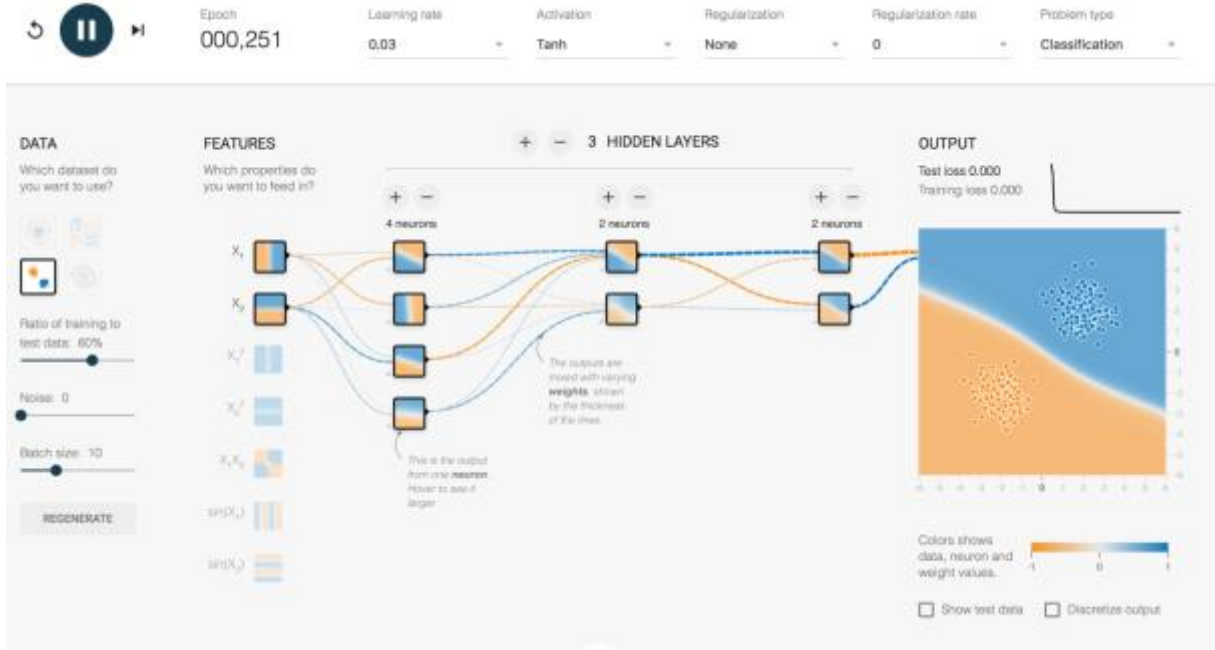


Şekil 2 – Makine Öğrenmesi

Makine öğrenmesi ya da makine öğrenimi temel olarak 2'ye ayrılır:

- **Gözetimli Öğrenme (Supervised Learning):** Makinenin sınıflandırılmış veya etiketlenilmiş veriler kullanılarak eğitilmesidir. Veriler etiketlenilmiş olduğu için sonuç değerlerinin doğruluğunun karşılaştırılmasında ve makinenin doğruluk yüzdesinin de belirlenmesinde kullanılabilir.
- **Gözetimsiz Öğrenme (Unsupervised Learning):** Gözetimli öğrenmenin tam aksine makinenin sınıflandırılmamış veya etiketlenilmemiş veriler kullanılarak eğitilmesidir.

Makine öğrenmesi ile ilgili bilgilerimi pekiştirdikten sonra bu alanda en çok kullanılan kütüphaneleri araştırdım ve Google’ın geliştirdiği açık kaynak kodlu (*open-source*) makine öğrenme kütüphanesi olan **TensorFlow**’u kullanmaya karar verdim. Python, C++, Java, Go gibi bir çok dilde kullanılabilmesi, aynı zamanda GPU desteğinin olması ve çok detaylı dökümantasyonlara sahip olması TensorFlow’u seçmemin temel sebepleri oldu. Python bilgimin diğer dillere kıyasla daha iyi olması nedeniyle denemelerimi **python** programlama dili ile yapmaya karar verdim.



**Şekil 3 – TensorFlow Playground Classification Probleminin Görselleştirilmesi**

TensorFlow’un Python modülünü kurmadan önce kendi sistemimin etkilenmemesi için stajdaki projelerimde kullanabileceğim bir *virtual environment* oluşturdum. Bilgisayarımda TensorFlow’un GPU desteği verdiği bir donanım olmadığı için sadece CPU’da çalışan versiyonunu kurdum.

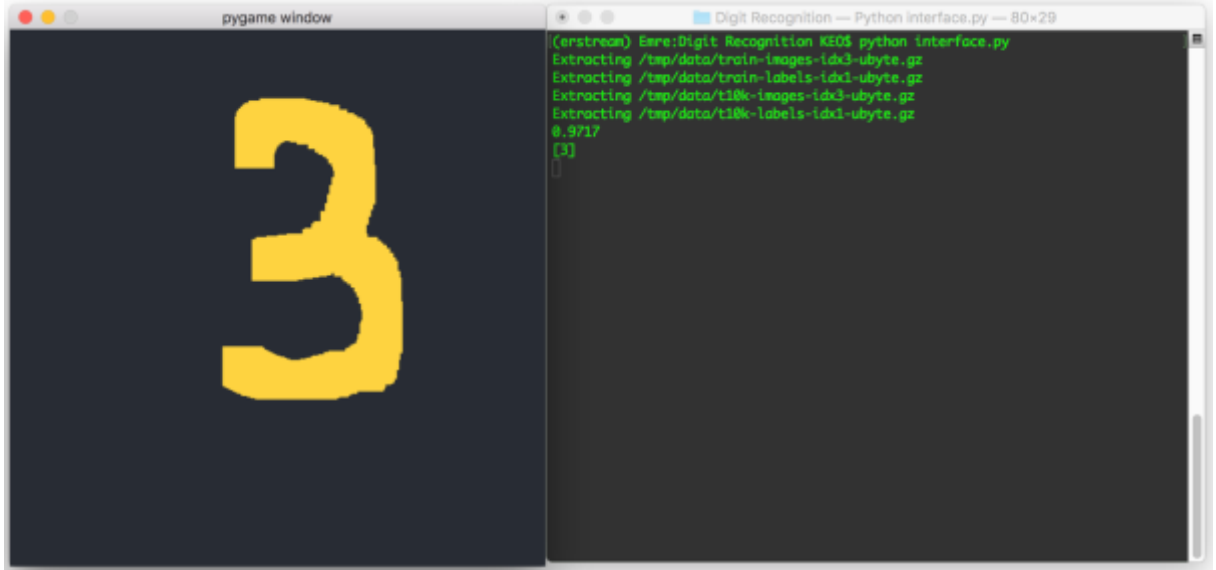
```
$ virtualenv --system-site-packages erstream          # virtual environmentin oluşturulması
$ source ~/erstream/bin/activate                    # virtual environmentin etkinleştirilmesi
(erstream)$ pip install --upgrade tensorflow         # tensorflow modülünün kurulması
```

**Şekil 4 – TensorFlow’un kurulumu**

Yeni bir programlama dilini öğrenirken ilginç bir gelenek vardır; ilk olarak ekrana nasıl “Hello world” yazdırılacağı öğrenilir. Makine öğrenmesinde de **MNIST** (multinomial logistic) gelenekselleşmiştir. MNIST bilgisayarın anlayabileceği bir veri kümesidir ve 70000’e yakın el yazısı ile yazılmış rakam resmi içermektedir.

Tensorflow bilgimi ve makine öğrenmesinin nasıl çalıştığını pekiştirmek için TensorFlow dökümanlarından faydalanarak **Digit Recognition** isimli uygulama geliştirdim. Python’un **PyGame** kütüphanesini kullanarak (arayüz işlemleri için) çizilen rakamın hangi rakam olduğunu tahmin etmeye çalıştım. Uygulamayı Python için avantajlı özellikleri olan **PyCharm** editöründe geliştirdim. Programa ait örnek kodları **Ek-1** ve **Ek-2** de bulabilirsiniz.





**Şekil 5** – Digit Recognition Programının Örnek Kullanımı

Digit Recognition programı ile birlikte makine öğrenmesi ve TensorFlow kütüphanesi ile ilgili bilgilerim oldukça gelişti. Bu aşamadan sonra asıl proje olan film öneri sistemleri hakkında araştırmaya başladım.

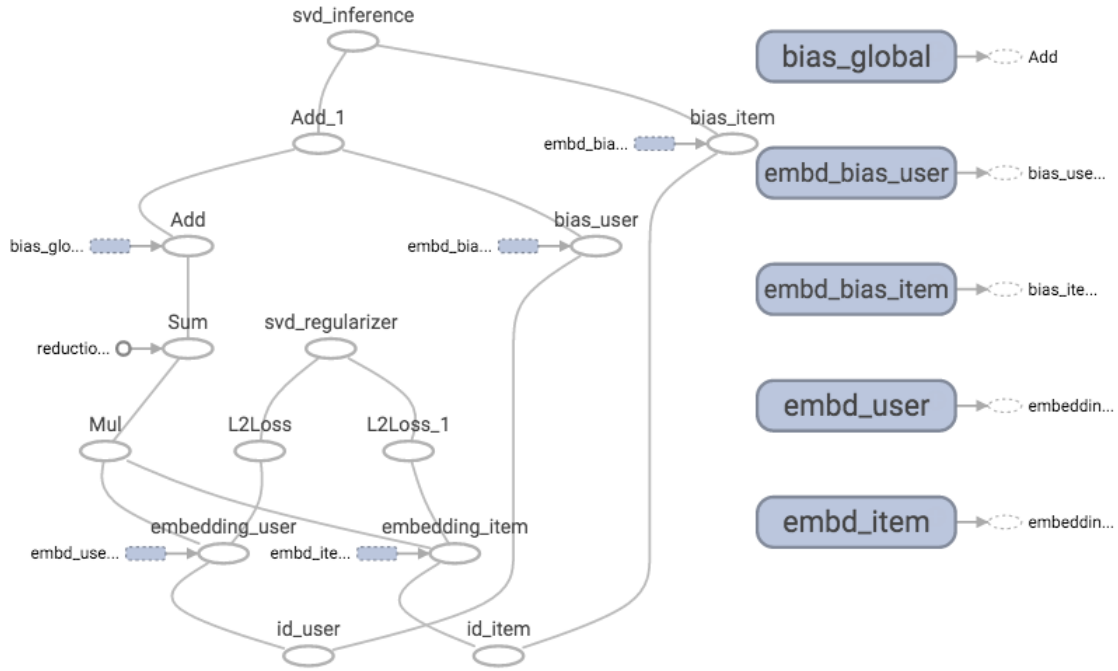
Öneri sistemlerinde kullanılan algoritmaları iki kısımda inceleme incelemek mümkün:

- **İçerik Bazlı Algoritmalar (*Content-base Algorithms*)**: Bu algoritmalarda filmler arasındaki ilişkiler hesaplanır ve bu ilişkiler kullanılarak yeni filmler önerilir.
- **İşbirliğine Dayalı Filtreleme (*Collaborative Filtering*)**: Bu algoritmalarda kullanıcı ile benzer filmleri izleyen kişiler bulunur ve onların izledikleri filmler kullanıcıya önerilir.

**YouTube** internet sitesinde yayınlanan Google Cloud Platform User Group Singapur konferansının videosunda [7] TensorFlow ile öneri sistemlerinin nasıl yazılabileceği ile ilgili fikir sabibi oldum ve **Şekil-6**'daki modeli inceledim.

## GELİŞTİRME

Araştırma kısmı bittikten sonra projeyi geliştirme süreci başladı. Bu süreçte araştırma aşamasında edindiğim **Şekil-6**'daki modeli kendim inşa ettim.



**Şekil 6 – Film Öneri Sistemi Modelinin Grafiği**

```
def model(self, dimension=10): # dimension represents number of genres
    with tf.device("/cpu:0"):
        bias_global = tf.get_variable("bias_global", shape=[])

        w_bias_user = tf.get_variable("embd_bias_user", shape=[self.max_user_count])
        w_bias_item = tf.get_variable("embd_bias_item", shape=[self.max_item_count])

        bias_user = tf.nn.embedding_lookup(w_bias_user, self.user_batch, name="bias_user")
        bias_item = tf.nn.embedding_lookup(w_bias_item, self.item_batch, name="bias_item")

        w_user = tf.get_variable("embd_user", shape=[self.max_user_count, dimension],
                                initializer=tf.truncated_normal_initializer(stddev=0.02))
        w_item = tf.get_variable("embd_item", shape=[self.max_item_count, dimension],
                                initializer=tf.truncated_normal_initializer(stddev=0.02))

        embd_user = tf.nn.embedding_lookup(w_user, self.user_batch, name="embedding_user")
        embd_item = tf.nn.embedding_lookup(w_item, self.item_batch, name="embedding_item")

        with tf.device(self.device):
            prediction = tf.reduce_sum(tf.multiply(embd_user, embd_item), 1)
            prediction = tf.add(prediction, bias_global)
            prediction = tf.add(prediction, bias_user)
            prediction = tf.add(prediction, bias_item, name="svd_inference")

            regularizer = tf.add(tf.nn.l2_loss(embd_user), tf.nn.l2_loss(embd_item), name="svd_regularizer")

    return prediction, regularizer
```

**Şekil 7 – Film Öneri Sistemi Modeli**

Kurduğum bu modeli eğitmek için **MovieLens** şirketinin sunduğu **kullanıcı, film, filme verilen puan** bilgilerini içeren verilerini kullandım. Veriler **csv** formatında sunulduğu için csv dosyaları için kullanışlı fonksiyonları olan pythondaki **pandas** modülü ile verileri okudum ve eğitme fonksiyonu olan **train** fonksiyonunda bu verileri kullandım. Bu aşamada 2 temel problem ile karşılaştım. İlki verilerin boyutunun çok yüksek olması nedeniyle verinin tamamını tek seferde train fonksiyonuna vermek mümkün değildi. Bu sorunu veriden rasgele daha küçük parçalar oluşturup onları train fonksiyonuna göndererek çözdüm. Karşılaştığım diğer sorun ise büyük csv dosyalarında pandas kütüphanesinin RAM’de çok fazla hafıza kullanmasından dolayı çok yavaş çalışması oldu. Bu sorunu da okuma işlemini yapan kodu kendim yazarak çözdüm. Train fonksiyonunu ve csv dosyasından veri okuma kodlarını **Ek-3** ve **Ek-4** de bulabilirsiniz.

TensorFlow modellerin eğitilmesi ve asıl öğrenme işlemini yapan bir çok **optimizer** hazır olarak sunmaktadır. Optimizerlar modellere ve eğitilirken kullanılan verilere göre birbirlerine üstünlük kurabilmekteler. Bu nedenle optimezerları tek tek test ettim ve hızlı ve daha iyi sonuç veren optimizerı bulmaya çalıştım. Yaptığım testlere göre FtrlOptimezer diğerlerinden daha hızlı çalışmakta va daha iyi sonuçlar üretmektedir.

Optimzerların kaybetme sayıları ve ne kadar süre çalıştığı aşağıda verilmektedir.

#### # FtrlOptimizer

Epoch: 1 Loss: 109730.610058  
Epoch: 2 Loss: 40779.2034035  
Epoch: 3 Loss: 37088.6556015  
Epoch: 4 Loss: 34623.9237347  
Epoch: 5 Loss: 33004.9797764  
Epoch: 6 Loss: 31948.9984455  
Epoch: 7 Loss: 31309.218195

real 0m52.811s  
user 0m53.772s  
sys 0m6.922s

#### # AdamOptimizer

Epoch: 1 Loss: 255564.115196  
Epoch: 2 Loss: 316349.247147  
Epoch: 3 Loss: 310102.635696  
Epoch: 4 Loss: 346084.011276  
Epoch: 5 Loss: 299993.485222  
Epoch: 6 Loss: 262621.729095  
Epoch: 7 Loss: 315639.295792

real 5m51.596s  
user 13m41.092s  
sys 3m16.993s

#### # RMSPropOptimizer

Epoch: 1 Loss: 47461.4948082  
Epoch: 2 Loss: 39880.5316582  
Epoch: 3 Loss: 38843.0034943  
Epoch: 4 Loss: 38345.0327778  
Epoch: 5 Loss: 37854.1822968  
Epoch: 6 Loss: 37422.8493271  
Epoch: 7 Loss: 37095.6983414

real 0m52.811s  
user 0m53.772s  
sys 0m6.922s

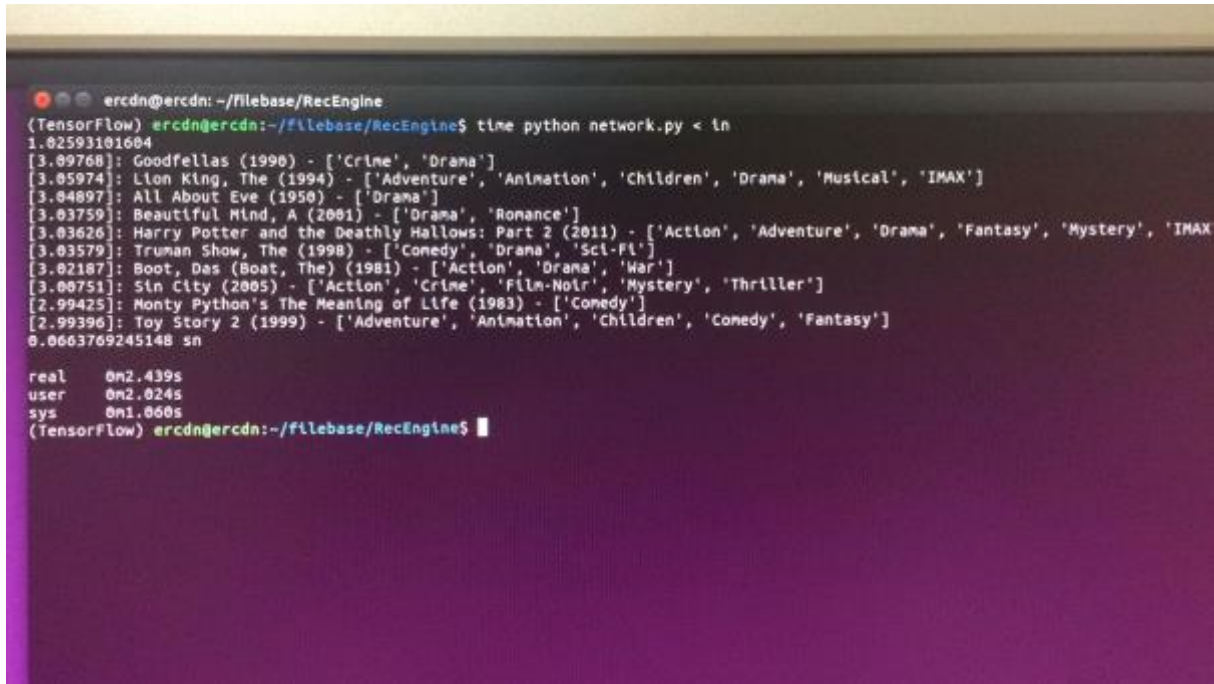
#### # AdagradOptimizer

Epoch: 1 Loss: 55707.7117214  
Epoch: 2 Loss: 44832.8309116  
Epoch: 3 Loss: 42404.8949413  
Epoch: 4 Loss: 41285.0513725  
Epoch: 5 Loss: 40045.0765572  
Epoch: 6 Loss: 39007.5849247  
Epoch: 7 Loss: 38583.1935673

real 0m51.885s  
user 0m53.299s  
sys 0m6.451s

**Şekil 8 – Optimizerların Sonuçları**

Eğitirken kullandığım veriler çok büyük olduğu için programın çalışma süresi oldukça uzun sürmekteydi. Bu sebeple şirketten TensorFlow’u GPU’da çalıştırabileceğim bir makine talep ettim ve programı bu ortamda test etmeye başladım. Ortalama bir GPU ile programda yaklaşık olarak 5 kat hızlanma gözlemlendi.



```
ercdn@ercdn: ~/Filebase/RecEngine
(TensorFlow) ercdn@ercdn:~/Filebase/RecEngine$ time python network.py < in
1.02593101604
[3.09768]: Goodfellas (1990) - ['Crime', 'Drama']
[3.05974]: Lion King, The (1994) - ['Adventure', 'Animation', 'Children', 'Drama', 'Musical', 'IMAX']
[3.04897]: All About Eve (1950) - ['Drama']
[3.03759]: Beautiful Mind, A (2001) - ['Drama', 'Romance']
[3.03626]: Harry Potter and the Deathly Hallows: Part 2 (2011) - ['Action', 'Adventure', 'Drama', 'Fantasy', 'Mystery', 'IMAX']
[3.03579]: Truman Show, The (1998) - ['Comedy', 'Drama', 'Sci-Fi']
[3.02187]: Boot, Das (Boat, The) (1981) - ['Action', 'Drama', 'War']
[3.00751]: Sin City (2005) - ['Action', 'Crime', 'Film-Noir', 'Mystery', 'Thriller']
[2.99425]: Monty Python's The Meaning of Life (1983) - ['Comedy']
[2.99396]: Toy Story 2 (1999) - ['Adventure', 'Animation', 'Children', 'Comedy', 'Fantasy']
0.0663769245148 sn

real    0m2.439s
user    0m2.024s
sys     0m1.060s
(TensorFlow) ercdn@ercdn:~/filebase/RecEngine$
```

Şekil 9 – Film Öneri Sistemi Örnek Çıktısı

## **SONUÇ**

Yaptığım bu stajda daha önceden çok merak ettiğim makine öğrenmesi hakkında detaylı bilgi edinmiş, örnek projeler geliştirmiş oldum. Dünya çapında bir çok şirketin kullandığı Google'ın makine öğrenmesi alanında geliştirdiği TensorFlow kütüphanesini inceleme ve proje yapma fırsatı bularak teknik anlamda gelişme fırsatı buldum.

Şirket ortamının nasıl olduğunu, sektöre yönelik ürünlerin ve hizmetlerin nasıl üretildiğini gözlemleyebilmek sektöre yönelik bilgimi arttırdı. Alanında uzman kişilerin yardımı ile araştırma-geliştirme projelerinde nasıl bir yol izlenmesi gerektiği hakkında çok değerli fikirler edindim. Sağladığı bu staj imkanından dolayı Ersteam şirketine teşekkür ederim.

## REFERANSLAR

1. *Experts System. Makine Öğrenmesi.* <http://www.expertsystem.com/machine-learning-definition/>
2. *Analytics Vidhya Makine Öğrenmesi.* <https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>
3. *Tecleer. Makine Öğrenmesi.* <https://www.techleer.com/articles/203-machine-learning-algorithm-backbone-of-emerging-technologies/>
4. *Şekil 2.* <https://dzone.com/articles/understanding-machine-learning>
5. *GroupLens.* <https://grouplens.org/datasets/movielens/>
6. *Google.* <http://tensorflow.com>
7. *Youtube. Recommendation Engine.* <https://www.youtube.com/watch?v=TNiWwaMGYzo>

## EKLER

**Ek 1:** El yazısı ile yazılmış rakamların tanınması projesindeki model ve MNIST verisi ile eğitilmesi

```
def model(self):
    sizes = [self.data_size] + self.layers + [self.classes]
    layers = []

    for i in range(1, len(sizes)):
        shape = [sizes[i - 1], sizes[i]]
        layer = (
            tf.Variable(tf.random_normal(shape), name='W'), # weights
            tf.Variable(tf.random_normal([sizes[i]]), name='b') # biases
        )
        layers.append(layer)

    results = [self.x]

    for W, b in layers[:-1]:
        layer = tf.add(tf.matmul(results[-1], W), b)
        layer = tf.nn.relu(layer) # activation
        results.append(layer)

    return tf.matmul(results[-1], layers[-1][0]) + layers[-1][1]

def train(self, epochs=10, save=False):
    with tf.device(self.device):
        cost = tf.reduce_mean(
            tf.nn.softmax_cross_entropy_with_logits(logits=self.prediction, labels=self.y)
        )
        optimizer = tf.train.AdamOptimizer().minimize(cost)

    self.session.run(tf.global_variables_initializer())

    for ep in range(epochs):
        loss = 0

        for _ in range(int(self.mnist.train.num_examples / self.batch_size)):
            x, y = self.mnist.train.next_batch(self.batch_size)
            _, c = self.session.run([optimizer, cost], {self.x: x, self.y: y})
            loss += c
        print(loss)

    if save:
        self.save()
```



**Ek 2:** El yazısı ile yazılmış rakamların tanınması projesindeki arayüzün PyGame ile oluşturulması

```
class InterFace(object):
    def __init__(self):
        self.scalar = 17
        self.screen = pygame.display.set_mode((28 * self.scalar, 28 * self.scalar))
        self.screen.fill(colors['bg'])
        pygame.display.flip()

        self.matrix = [[0]*28 for _ in range(28)]

    def recognize(self):
        query = [reduce(lambda x, y: x + y, self.matrix)]
        return network.check(query)

    def draw(self, x, y):
        t = self.scalar
        for i in range(-t, t+1):
            for j in range(-t, t+1):
                self.matrix[(y+i) // self.scalar][(x+j) // self.scalar] = 1

        for i in range(x - t, x + t + 1):
            for j in range(y - t, y + t + 1):
                self.screen.set_at((i, j), colors['text'])

        pygame.display.flip()

    def clear(self):
        self.screen.fill(colors['bg'])

        for i in range(28):
            for j in range(28):
                self.matrix[i][j] = 0

        pygame.display.flip()

    def main(self):
        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()

                if pygame.mouse.get_pressed()[0]:
                    try:
                        self.draw(*event.pos)
                    except Exception:
                        pass

                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_r:
                        print(self.recognize())

                    if event.key == pygame.K_c:
                        self.clear()
```



### Ek-3: Film öneri sisteminde kullanılan train fonksiyonu

```
def train(self, epochs=10, learning_rate=0.1, save=None):
    with tf.device(self.device):
        cost_l2 = tf.nn.l2_loss(tf.subtract(self.prediction, self.rate_batch))
        penalty = tf.constant(0.1, dtype=tf.float32, shape=[], name="l2")

        cost = tf.add(cost_l2, tf.multiply(self.regularizer, penalty))
        optimizer = tf.train.AdagradOptimizer(learning_rate).minimize(cost)

    self.session.run(tf.global_variables_initializer())
    training_data = Reader('train-ratings-s.csv', random=True)

    for ep in xrange(epochs):
        loss = 0
        for users, items, rates in training_data:
            _, c = self.session.run([optimizer, cost], {self.user_batch: users,
                                                         self.item_batch: items,
                                                         self.rate_batch: rates})
            loss += c

        training_data.iteration_count = 0
        print('Epoch:', ep+1, 'Loss:', loss)

    if save is not None:
        dirname = os.path.dirname(save)

        if not os.path.isdir(dirname):
            os.mkdir(dirname)

        self.saver.save(self.session, save)
```

#### Ek-4: Film öneri sisteminde kullanılan csv dosyasını okuyan yapı

```
class Reader(object):
    """generates an generator"""
    def __init__(self, path, random=False):
        with open(path, 'r') as stream:
            self.lines = [line.split(',') for line in stream]

        self.cursor = 0
        self.random = random
        self.num_examples = len(self.lines)

        self.default_iteration_size = 100

        self.iteration_count = 0
        self.iteration_limit = self.num_examples / self.default_iteration_size

    def next_batch(self, size=100):
        if size > self.num_examples \
            or self.cursor >= self.num_examples \
            or self.iteration_count >= self.iteration_limit:
            raise StopIteration

        if self.random:
            indexes = sample(range(self.num_examples), size)
        else:
            indexes = range(self.cursor, min(self.cursor + size, self.num_examples))
            self.cursor += size

        users = []
        items = []
        rates = []

        for i in indexes:
            users.append(int(self.lines[i][0]))
            items.append(int(self.lines[i][1]))
            rates.append(float(self.lines[i][2]))

        self.iteration_count += 1
        return users, items, rates

    def __iter__(self):
        return self

    def next(self):
        return self.next_batch()

    def __len__(self):
        return self.num_examples
```