# Giriş:

Farklı makine öğrenmesi yöntemleri ve yapay sinir ağları kullanılarak İnternet sitesi sınıflandırma problemi incelenmiştir. Bu sınıflandırma probleminin çözümü için, İkili Sınıflandırma ve Çok Sınıflı Sınıflandırma olarak iki farklı yaklaşım uygulanmış,

Her iki yaklaşım da çalışma kapsamında toplanan İnternet siteleri üzerinde test edilip, performansları karşılaştırılmıştır. Başarıma bakıldığında,

* İkili sınıflandırıcılar için en iyi performans gösteren algoritma **Lojistik Regresyon** olmuştur.
* Çok Sınıflı Sınıflandırma yaklaşımında uygulanan algoritmalar arasından ise en yüksek başarıma sahip yöntem **Destek Vektör Makineleri (Support Vector Machines, SVM)** olmuştur.

Çok Sınıflı Sınıflandırma problemi için farklı kelime vektörleştirme yöntemleri denenmiş ve performansları karşılaştırılmıştır. İkili ve Çok Sınıflı sınıflandırma yaklaşımlarında kullanılan algoritmaların ayrı ayrı ve farklı vektörleştirme yöntemleri ile denenmesi, İnternet sayfalarının sınıflandırılması ve içerik filtrelenmesi problemlerinin birlikte ele alınmıştır. Öğrenme yöntemlerinin öğrenme ve test setlerinin yanlılığını araştırmak için performans araçlarından **F1 Skoru, hata matrisi** gibi teknikler kullanılmıştır.

İkili Sınıflandırmada kullanılan yöntemlerin, analizin süreçleri boyunca işlemsel performans (süre) göz önüne alındığında, **Lojistik Regresyon ve Bernoulli Naive Bayes** sınıflanırıcılarının, **yapay sinir ağlarına** göre **150 kat daha hızlı** sonuçlandığı gözlenmiştir.

## Metin Sınıflandırma:

Metin sınıflandırma, içeriğine göre metne etiket veya kategori atama işlemi olarak tanımlanabilir. Metin sınıflandırması, matematiksel olarak Denklem 1 [8] şeklinde ifade edilebilir.

𝑌 = 𝑓(𝑋, 𝜃) + 𝜀 (1)

Bu denklemde,

***f*:** eğitim verilerini kullanarak tahmin eden sınıflandırıcı veya sınıflandırma modeli

***Y*:** metnin belirli bir sınıfa üyeliğini belirten sayısal değer

***X*:** kelimeler veya kelime gruplarını içeren bir metin vektörü

***θ*:** f fonksiyonuyla ilişkilendirilen bilinmeyen parametrelerin kümesi

***ε*:** sınıflandırma hatasıdır.

## Kelime Vektörleştirme Algoritmaları

Kelime vektörleştirme, makine ve derin öğrenme süreçlerinden önce uygulanan bir dönüşüm işlemidir. Bu dönüşüm ile birlikte metin farklı yöntemlere göre kendi sınıfı içerisinde sayısal veriler içeren vektörlere dönüştürülür. Bu işlemin sonucunda, metin verisi algoritmalar tarafından eğitilebilecek ve analiz edilebilecek sayısal değerlere dönüşmüş olur. Bu yöntemlerin bazılarını kullanırken dikkatli olunması gerekir. Çünkü kategorilere farkında olmadan birbirleri arasında üstünlük verebilecek değerler atanabilir. Bu yüzden metin sınıflandırmasının başarımını etkileyecek en önemli süreçlerden biri de kelime ve kelime gruplarının vektörleştirilmesidir.

## Sınıflandırma Sırasında Kullanılan Algoritmalar

Gözetimli öğrenme algoritmaları, önceden etiketlenmiş verileri girdi olarak almaktadır. Metin sınıflandırma probleminde, İnternet siteleri ait olduğu sınıflarla birlikte modele beslenmektedir. Böylelikle model eğitim setindeki sınıflandırmayı öğrenebilmektedir.

* **İkili Sınıflandırma:** İkili veya binom sınıflandırma, belirli bir veri setinin öğelerini bir sınıflandırma kuralı temelinde iki gruptan (her birinin hangi gruba ait olduğunu tahmin ederek) birine atama işlemidir. Bir tür gözetimli öğrenme olan bu yöntemde, eğitim sırasında özellik seti ile birlikte etiketlenen değişken, modeli oluşturulan sınıf için 1, geri kalan örnekler için 0 değeri alır [16]. Örneğin “Araba” sınıfı için bir ikili sınıflandırma modeli oluşturulmak istendiğinde, “Araba” sınıfına ait İnternet sayfaları 1, geri kalan sayfalar 0 olarak etiketlenir. Bu çalışmada, tanımlanmış sınıf sayısı kadar (63 tane) sınıflandırıcı oluşturulmuş ve bunların her birinin başarım oranları, hata matrisleri ve işlem süreleri çıkarılmıştır. İkili sınıflandırma sürecinde yapılan her bir işlem Şekil 2a’da gösterilmiştir. Tam Bağlantılı Yapay Sinir Ağları (Fully Connected Neural Network, FCNN), Lojistik Regresyon (Logistic Regression) ve Bernoulli Naive Bayes algoritmaları ikili sınıflandırma için ayrı ayrı kullanılıp her birinin başarım oranları incelenmiştir [17].
* **Çok Sınıflı Sınıflandırma:** Çok sınıflı sınıflandırma yöntemi bir gözetimli sınıflandırma yöntemi olup, bu sınıflandırma içinde bulunan algoritmalar üç ve daha fazla sınıfı olan veriler için kullanılmaktadır. Çalışmada sınıflandırma modeli oluşturulurken, her bir İnternet sitesi sınıfı, kendi özellik setinde (kelime vektörü) temsil edilecek şekilde eğitilmiştir. Eğitim sonrası algoritmaların testleri sürecinde de yine tahmin edilmesi beklenen çıktı yine her bir 63 sınıftan biridir. İkili sınıflandırmadan farklı olarak; her bir sınıf için ayrı ayrı sınıflandırıcı yerine, tek bir sınıflandırıcı oluşturulup, tüm testler bu model üzerinden yapılmıştır. Çok sınıflı sınıflandırma sürecinde yapılan her bir işlem Şekil 2b’de gösterilmiştir. Çok sınıflı sınıflandırma modelini oluşturmak için, Multinominal Naive Bayes, Rastgele Orman (Random Forest) ve Destek Vektör Makineleri (Support Vector Machines – SVM) sınıflandırıcıları ayrı ayrı kullanılıp her birinin başarım oranları incelenmiştir [18] [19] [20].



Şekil 2. Sınıflandırma Süreçleri: (a) İkili Sınıflandırma Süreci (b) Çok Sınıflı Sınıflandırma Süreci

# Benzer Çalışmalar

## Metin Sınıflandırma Çalışmaları

Rong, Yuancheng ve Xiangqian yaptığı çalışmada, ilk kez derin inanç ağları (deep belief networks) kullanılmıştır.

Agren ve Agren, doğal dil işleme (Natural Language Process – NLP) ile metnin yazarının; yanlı, yansız ya da tarafsız olup olmadığını; haber başlığının haber içeriği ile ilişkisi ile ilgili çıkarımlarda bulmaya çalışılmıştır.

Kudugunta ve Ferrara, bir yazılım ile otomatik oluşturulan tweetleri tespit etmek amaçlamıştır. Çalışma kapsamında tekrarlayan sinir ağlarının bir türü olan uzun-kısa vadeli hafıza ağları kullanılmıştır.

Figueira çalışmasında da; ayrılmış, yalan haberlerin tespitinde ise gizli Markov modelleri kullanılmıştır.

Yao ve Zhi-Min, modern yazı sınıflandırma metotlarının konu bazlı yazı sınıflandırmasında eksik kalabildiğinin altını çizmekte ve DNB adını verdikleri metodunu Destek Vektör Makineleri (Support Vector Machine - SVM), *k*-En Yakın Komşu (*k* Nearest Neighbour – KNN), Naive Bayes gibi yöntemler ile kıyaslamakta olup, DNB'nin sınıflandırma sonuçlarının başarısını arttırdığını vurgulamaktadır.

Trstenjak, Mikac ve Donko yaptıkları çalışmada [24], KNN algoritmasının metin sınıflandırma için çok popüler bir yöntem olduğunu vurgulayıp, TF-IDF ile KNN algoritmasını denemiştir.

Liang ve arkadaşları, sözlük terimlerini bulmak için belgelerde otomatik bölümlendirme işlemi gizli anlamsal indeksleme kullanılarak yapılmış ve ardından metin sınıflandırması algoritması olarak KNN kullanılmıştır. Ek bir sözlük kullanmanın etkileri ve verimliliği ele alınmış, ayrıca farklı veri toplama yöntemleri denenmiş ve sınıflandırma performansını geliştirmede rol oynayan yeni bir oylama yöntemi önerilmiştir. Deneysel sonuçlar; önerilen yaklaşımın, geleneksel sınıflandırma yöntemine göre üstün bir performans göstermiştir.

Kotevska, Padi ve Llbath, Twitter verileri kullanılarak NLP ve Rastgele Orman sınıflandırıcısının birlikte kullanıldığı bir otomatik sınıflandırma yöntemi önerilmiştir. Rastgele Orman sınıflandırıcısını kullanarak bu tweetler otomatik olarak önceden tanımlanmış kategorilere ayrıştırılmıştır. Rastgele Orman sınıflandırıcısının SVM ve Naive Bayes sınıflandırıcılarından daha iyi performans ortaya koyduğu gösterilmiştir.

Chen, Cheng ve Cheng, öğrenme temelli Uzun ve Kısa Vadeli Hafıza Ağları (Long-Short Term Memory-LSTM) kullanmıştır.

Qazi ve Guadar, önermektedir. Ontoloji, herhangi bir alan için bilgi temsilinin merkezini oluşturur. Önerilen yaklaşım etki alanı ontolojisini oluşturur ve tahmin performansını önemli ölçüde artıran özellikleri seçmektedir. Alan tabanlı İnternet sayfalarında deneyler yapılmış ve sınıflandırma performansı, teknik sınıflandırma algoritmalarıyla hesaplanmıştır. Deneysel analizler, önerilen yaklaşımın geleneksel anahtar kelime temelli yaklaşımlara kıyasla çok daha iyi sonuçlar verdiğini göstermektedir.

## İnternet Sitesi Sınıflandırma Çalışmaları

Cuzzola ve arkadaşları, çok sınıflı sınıflandırma yerine, ikili sınıflandırma teknikleri kullanmışlardır. Bu teknikler; Sinir ağları sınıflayıcısı ve destek vektör makineleridir. Yazarlar, verdikleri deneysel sonuçlara göre sinir ağlarının daha iyi sınıflandırma performansı göstermektedir. İnternet sitesi sınıflandırma problemi için gözetimsiz sınıflandırma teknikleri kullanılmıştır.

Li ve arkadaşları çalışmasında, İnternet sayfasındakı kelimeler (“N-gram with Wikipedia Entity Words” veri seti kullanılarak) ana metinden, başlıktan çıkarılıp Bayes sınıflandırıcısı ile tahmin edilmeye çalışılmıştır.

Lee, Yeh ve Chuang, en iyi ağırlıkları benimsemeyen yeni bir basitleştirilmiş sürüler optimizasyonu (Simplified Swarm Optimization – SSO) önerilmektedir. Ayrıca, parametre ayarları, SSO'nun güncelleme mekanizmasında önemli bir rol oynar, böylece parametre ayarlarını belirlemek için bir Taguchi yöntemini kullanılmıştır. Algoritmanın etkililiğini göstermek için, performansını iyi bilinen Genetik Algoritma (GA), Bayes sınıflandırıcı ve KNN sınıflandırıcıları ile dört veri kümesine göre karşılaştırılmıştır.

Gali, Meriescu-Istodor ve Fränti, İnternet sayfasında görüntülenen reklamlardan ötürü yanıltıcı olabileceğini öne sürmüştür ve bu sorunun çözümü için istatistiksel özellikler, dil bilgisi, metin bölümleme metotlarını belirtmiştir. Naive Bayes, KNN, SVM gibi yöntemlerle değerlendirilen bu yaklaşımın, metin analizi ve gruplandırılması açısından önem taşıdığı vurgulanmaktadır.

**NOT:** Veri ön işlemesi (data preprocessing), [ilkelleme (regression)](http://www.bilgisayarkavramlari.com/2013/01/26/linear-regression-dogrusal-ilkelleme/), sınıflandırma (classification), gruplandırma (clustering), [özellik seçimi veya özellik çıkarımı (feature extraction)](http://www.bilgisayarkavramlari.com/2008/12/01/ozellik-cikarimi-feature-extraction/) bunlardan bazılarıdır.

# Machinelearningmastery

ML algoritmaları kategorik veriler ile doğrudan çalışmaz. Kategorik verilerin sayısal verilere dönüştürülmesi gerekir. Bu, bir dizi sınıflandırma tipi problemi ile çalışırken ve Uzun Kısa Süreli Bellek tekrarlayan sinir ağları gibi derin öğrenme yöntemlerini kullanmayı planladığınızda geçerlidir. One Hot Encoding (OHE) kategorik verilerin ikili (0-1) vektörler ile ifade edilmesidir. İlk olarak kategorik verilerin tamsayılar üzerinde haritalanması gerekir. Ardından her tamsayı, tamsayının indexi haricindeki bütün verilen 0 olduğu bir ikili vektör olarak ifade edilir. OHE kategorik verileri daha anlamlı olarak ifade etmemizi sağlar. Bir çok ML algoritması kategorik veriler ile doğrudan çalışmaz.

# Geleneksel Kelime Temsil Yöntemleri

Kelime hazinesinin (lügat) oluşturulması. N-grams kelime hazinesinin kalıplar, deyimler gibi eklentiler ile genişletilmesi. Kelime hazinesinin azaltılması. Örneğin, büyük küçük harf ayrımının kaldırılması, noktalama işaretlerinini elenmesi, kelime köklerine göre farklılıkların kaldırılması(stemming), oynuyordum oynadım, yanlış yazılmış kelimelerin elenmesi, Frekansı çok geçen yardımcı kelimlerin tf-idf yöntemleri kullanılarak elenmesi vb.

Vocabulary: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.

Sparsity: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to harness so little information in such a large representational space.

Meaning: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

## Bag of Words

## One Hot Encoding

## Tf-Idf

## Performance

The training speed can be significantly improved by using parallel training on multiple-CPU machine (use the switch '-threads N'). The hyper-parameter choice is crucial for performance (both speed and accuracy), however varies for different applications. The main choices to make are:

* architecture: skip-gram (slower, better for infrequent words) vs CBOW (fast)
* the training algorithm: hierarchical softmax (better for infrequent words) vs negative sampling (better for frequent words, better with low dimensional vectors)
* sub-sampling of frequent words: can improve both accuracy and speed for large data sets (useful values are in range 1e-3 to 1e-5)
* dimensionality of the word vectors: usually more is better, but not always
* context (window) size: for skip-gram usually around 10, for CBOW around 5

# Sınıflandırma

## Bayes Sınıflandırma Algoritması

Standard Naive Bayes only supports categorical features, while Gaussian Naive Bayes only supports continuously valued features. If your data set's features are categorical, standard Naive Bayes is the better choice. If your data set's features are continuous, Gaussian Naive Bayes may be better. If your data set contains both categorical and continuous features, you have two options. You can either discretize your continuous features via bucketing or a similar methodology, or you can use a hybrid Naive Bayes model. Unfortunately none of the standard machine learning libraries seem to include such a model. If you're looking for more information about hybrid Naive Bayes models, check out:

1. <https://github.com/remykarem/mixed-naive-bayes> ,
2. <https://stackoverflow.com/questions/14254203/mixing-categorial-and-continuous-data-in-naive-bayes-classifier-using-scikit-lea>

**Bernoulli Naive Bayes:** It assumes that all our features are binary such that they take only two values. Means 0s can represent “word does not occur in the document” and 1s as "word occurs in the document". Bernoulli Naive Bayes is based on the Bernoulli distribution based on discrete distribution.

**Multinomial Naive Bayes:** Its is used when we have discrete data (e.g. movie ratings ranging 1 and 5 as each rating will have certain frequency to represent). In text learning we have the count of each word to predict the class or label.

**Gaussian Naive Bayes:**  Because of the assumption of the normal distribution, Gaussian Naive Bayes is used in cases when all our features are continuous. For example in Iris dataset features are sepal width, petal width, sepal length, petal length. So its features can have different values in data set as width and length can vary. We can’t represent features in terms of their occurrences. This means data is continuous. Hence we use Gaussian Naive Bayes here.  Here, we assume that the features follow a normal distribution. Instead of discrete counts, we have continuous features (e.g., the popular Iris dataset where the features are sepal width, petal width, sepal length, petal length).

**GaussianNB:** Tahmin edeceğiniz veri veya kolon sürekli (real,ondalıklı vs.) ise

**BernoulliNB:** Tahmin edeceğiniz veri veya kolon ikili ise ( Evet/Hayır , Sigara içiyor/ İçmiyor vs.)

**MultinomialNB:** Tahmin edeceğiniz veri veya kolon nominal ise ( Int sayılar ). Duruma göre bu üç sınıftan birini seçebilirsiniz. Modelin başarı durumunu etkiler.

## Lineer Regresyon Sınıflandırma Algoritması

## Ağaç Tabanlı Sınıflandırma Algoritmaları

### Light GBM

Light GBM is a gradient boosting framework that uses tree based learning algorithm.

* How it differs from other tree based algorithm?

Light GBM grows tree vertically while other algorithm grows trees horizontally meaning that Light GBM grows tree leaf-wise while other algorithm grows level-wise. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm. (Light GBM dengesiz ağaç, Diğerleri dengeli ağaç yapısı kullanır.)

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as ‘Light’ because of its high speed. Light GBM can handle the large size of data and takes lower memory to run. Another reason of why Light GBM is popular is because it focuses on accuracy of results. LGBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development. It is not advisable to use LGBM on small datasets. Light GBM is sensitive to overfitting and can easily overfit small data. Their is no threshold on the number of rows but my experience suggests me to use it only for data with 10,000+ rows. Implementation of Light GBM is easy, the only complicated thing is parameter tuning. Light GBM covers more than 100 parameters but don’t worry, you don’t need to learn all.

***Control Parameters***

1. **max\_depth:** It describes the maximum depth of tree. This parameter is used to handle model overfitting. Any time you feel that your model is overfitted, my first advice will be to lower max\_depth.
2. **min\_data\_in\_leaf:** It is the minimum number of the records a leaf may have. The default value is 20, optimum value. It is also used to deal over fitting.
3. **feature\_fraction:** Used when your boosting (discussed later) is random forest. 0.8 feature fraction means LightGBM will select 80% of parameters randomly in each iteration for building trees.
4. **bagging\_fraction:**specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid overfitting.
5. **early\_stopping\_round:** This parameter can help you speed up your analysis. Model will stop training if one metric of one validation data doesn’t improve in last early\_stopping\_round rounds. This will reduce excessive iterations.
6. **lambda:** lambda specifies regularization. Typical value ranges from 0 to 1.
7. **min\_gain\_to\_split:** This parameter will describe the minimum gain to make a split. It can used to control number of useful splits in tree.
8. **max\_cat\_group:** When the number of category is large, finding the split point on it is easily over-fitting. So LightGBM merges them into ‘max\_cat\_group’ groups, and finds the split points on the group boundaries, default:64

***Core Parameters***

1. **Task:** It specifies the task you want to perform on data. It may be either train or predict.
2. **application:** This is the most important parameter and specifies the application of your model, whether it is a regression problem or classification problem. LightGBM will by default consider model as a regression model. There is an important difference between classification and regression problems. That classification is the problem of predicting a discrete class label output for an example. That regression is the problem of predicting a continuous quantity output for an example.
3. **regression:** for regression
4. **binary:** for binary classification
5. **multiclass:** for multiclass classification problem
6. **boosting:** defines the type of algorithm you want to run, default=gdbt
7. **gbdt:** traditional Gradient Boosting Decision Tree. (Gradient boosting is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [regression](https://en.wikipedia.org/wiki/Regression_(machine_learning)) and [classification](https://en.wikipedia.org/wiki/Classification_(machine_learning)) problems, which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_(machine_learning)) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).)
8. **rf:** random forest
9. **dart:** Dropouts meet Multiple Additive Regression Trees
10. **goss:** Gradient-based One-Side Sampling
11. **num\_boost\_round:** Number of boosting iterations, typically 100+
12. **learning\_rate:** This determines the impact of each tree on the final outcome. GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. Typical values: 0.1, 0.001, 0.003…
13. **num\_leaves:** number of leaves in full tree, default: 31
14. **device:** default: cpu, can also pass gpu

***Metric parameter***

1. **metric:** again one of the important parameter as it specifies loss for model building. Below are few general losses for regression and classification.
2. **mae:** mean absolute error
3. **mse:** mean squared error
4. **binary\_logloss:** loss for binary classification
5. **multi\_logloss:** loss for multi classification

***IO parameter***

1. **max\_bin:** it denotes the maximum number of bin that feature value will bucket in.
2. **categorical\_feature:** It denotes the index of categorical features. If categorical\_features=0,1,2 then column 0, column 1 and column 2 are categorical variables.
3. **ignore\_column:** same as categorical\_features just instead of considering specific columns as categorical, it will completely ignore them.
4. **save\_binary:** If you are really dealing with the memory size of your data file then specify this parameter as ‘True’. Specifying parameter true will save the dataset to binary file, this binary file will speed your data reading time for the next time.

Knowing and using above parameters will definitely help you implement the model. Remember I said that implementation of LightGBM is easy but parameter tuning is difficult. So let’s first start with implementation and then I will give idea about the parameter tuning.