Yıldız Teknik Üniversitesi

Elektrik Elektronik Fakültesi
Bilgisayar Mühendisliği Bölümü

Yapay Zekâ 1. Ödevi

20011024 – Sait Yalçın

20011901 – Muhammed Kayra Bulut

sait.yalcin@std.yildiz.edu.tr

kayra.bulut@std.yildiz.edu.tr

1 Giriş

Makine öğrenmesi nedir?

Makine öğrenmesi, bilgisayarların veri tabanlarından belirli bir algoritma veya yöntem kullanarak öğrenmesini sağlayan bir alandır. Bu algoritma veya yöntemler, veriye dayalı olarak öğrenir ve sonuçları belirli bir görevi gerçekleştirmek için kullanılır.

Örneğin, bir sınıflandırma modeli oluşturmak istediğinizi varsayalım. Bu model, veri setinizdeki örneklerin özelliklerine dayalı olarak bir sınıfa ait olup olmadığını tahmin edebilir. Makine öğrenmesi kullanarak, veri setinizi eğitebilir ve modelinizi oluşturabilirsiniz. Ardından, bu modeli kullanarak, özelliklerini verdiğiniz bir örneğin hangi sınıfa ait olduğunu tahmin edebilirsiniz.

Bunun yanı sıra, regresyon modelleri ile de örneğin hava sıcaklığı, hisse senetleri fiyatları veya ev fiyatları gibi sürekli değişkenlerin tahminini yapabilirsiniz.

Makine Öğrenmesi Modellerinden Bazıları:

1-) K En Yakın Komşuluk:

KNN (K-Nearest Neighbors), sınıflandırma ve regresyon problemlerinde kullanılan bir makine öğrenmesi algoritmasıdır. Sınıflandırma problemlerinde verilen bir örneği en yakın komşularının sınıfıyla etiketleyerek sınıflandırma yapar. Regresyon problemlerinde ise verilen bir örneğin en yakın komşularının ortalama değerini kullanarak tahmin yapar.

Örnek olarak, bir restoranın müşteri verileriyle ilgili bir sınıflandırma problemini ele alalım. Müşterilerin özellikleri (yaş, cinsiyet, maddi durum, vb.) verilerek, bir müşterinin restorana geliş nedeni (kahvaltı, öğle yemeği, akşam yemeği) tahmin edilmek isteniyor. Bu durumda, KNN algoritması, müşterinin özellikleriyle en yakın komşularının geliş nedenlerini inceleyerek, müşterinin geliş nedenini tahmin edebilir.

2-) Destek Vektör Makineleri:

SVM (Support Vector Machine), özellikle sınıflandırma ve regresyon analizinde kullanılan bir makine öğrenmesi yöntemidir. SVM, veri noktalarının n-boyutlu uzayda temsil edildiği bir problemde, verileri birbirlerinden ayırmak için bir hiper düzlem oluşturarak sınıflandırma yapar. SVM'nin amacı, sınıflar arasında en geniş boşluğu bulan bir hiper düzlem oluşturmaktır. Bu boşluk, sınıfların doğru bir şekilde ayrılabilmesi için gereklidir. SVM, veri kümesindeki her bir örnek için bir özellik vektörü oluşturur ve bu özellik vektörlerini kullanarak öğrenme işlemini gerçekleştirir. SVM, özellikle veri setleri lineer olarak ayrılamadığında veya veri noktalarının sayısı çok fazla olduğunda etkili bir yöntemdir.

3-) Multinomial Naive Bayes:

Multinomial Naive Bayes (MNB), Naive Bayes algoritmalarının bir türüdür ve genellikle çok sınıflı sınıflandırma problemlerinde kullanılır. Özellikle doğal dil işlemede (NLP) sıklıkla kullanılır. MNB, bir belgenin sınıfını tahmin etmek için, belgenin içerdiği kelimelerin frekanslarını kullanır. Bu nedenle, MNB, bir belgeyi bir vektöre dönüştürür ve sınıflandırma işlemi vektör üzerinden yapılır. MNB, Naive

Bayes algoritmalarının diğer türleri gibi, sınıflandırma işlemini yapmak için Bayes teoremini kullanır. MNB, adını çoklu kategorik (multinomial) dağılımdan almıştır, çünkü kelime frekanslarını sayarak oluşturduğu vektörler, çoklu kategorik dağılımın bir örneğidir.

4-) Rastgele Orman:

RandomForest, birden fazla karar ağacını kullanarak bir veri kümesini sınıflandıran bir makine öğrenmesi algoritmasıdır. Karar ağaçları, veri kümesindeki özellikleri kullanarak birçok karar ağacı oluşturur ve her biri sonucu tahmin etmek için kullanılır. Sonuçlar, tüm karar ağaçlarının tahminlerinin ortalaması veya çoğunluğu alınarak elde edilir. Bu yöntem, veri kümesindeki gürültülü veya eksik verileri ele alır ve doğru sonuçları üretmek için birden fazla karar ağacının avantajını kullanır. Ayrıca, overfitting problemini de azaltır. Bu nedenle, RandomForest sınıflandırma ve regresyon problemlerinde yaygın olarak kullanılan bir algoritmadır.

5-) Karar Ağaçları:

Decision Tree, sınıflandırma ve regresyon problemleri için kullanılan, karar verme ağaçlarını oluşturarak veri setlerini analiz etmek ve sonuç tahminleri yapmak için kullanılan bir makine öğrenmesi algoritmasıdır. Bu algoritma, belirli bir hedef değişkeni tahmin etmek için veri kümesindeki özellikleri kullanarak bir ağaç yapısı oluşturur. Ağaç yapısı, her düğümdeki bir kararla birlikte bir dalga ayrılır ve bu kararlar sonunda sonuç değişkenine ulaşılır. Ağaç yapısı, verilerin özelliklerini değerlendirerek, verilerin bir sınıfa veya değere atanmasını sağlar. Decision Tree, verilerin anlaşılması için oldukça kullanışlı bir yöntemdir ve yüksek doğruluk oranları ile sonuçlar verir.

Vektörize Yöntemleri

TF-IDF (Term Frequency-Inverse Document Frequency) ve CountVectorizer, metin verilerini sayısal verilere dönüştürmek için kullanılan iki popüler vektörleştirme yöntemidir.

CountVectorizer, verilen belgelerdeki kelime frekanslarını sayar ve her kelime için bir sütun oluşturur. Bu yöntemde, sadece kelime sayıları dikkate alınır ve kelimenin belgedeki diğer kelimelerle ilişkisi göz ardı edilir. Örneğin, "the cat in the hat" cümlesi "the", "cat", "in", "the", ve "hat" olarak vektörize edilir.

TF-IDF, CountVectorizer'ın temel fikirlerinden yola çıkarak, bir kelimenin bir belgedeki önemini hesaplamak için bir ağırlık verir. Bir kelimenin sıklığı, o kelimenin belgedeki önemine göre azaltılır ve ayrıca o kelimenin diğer belgelerdeki sıklığı da dikkate alınır. Bu şekilde, çok yaygın olan kelimeler, daha az önemli olduğu düşünülür ve belgeyi tanımlayan özel kelimeler daha fazla önem kazanır.

Örneğin, bir belgenin "the cat in the hat" olduğunu varsayalım. CountVectorizer ile bu belge "the", "cat", "in", "the", ve "hat" şeklinde vektörize edilecektir. Ancak TF-IDF ile, örneğin "cat" kelimesi, belgenin özgünlüğünü ve önemini daha iyi yansıtmak için daha yüksek bir ağırlığa sahip olabilir.

Kısacası, CountVectorizer, belgedeki kelime sayılarına dayanırken, TF-IDF, bir kelimenin bir belgedeki önemini ve diğer belgelerdeki sıklığını da hesaba katarak belgeyi vektörize eder.

2 Projemiz

Projemizde amazon alışveriş sitesi yardımıyla elde ettiğimiz verileri, RandomForestClassifier, DecisionTreeClassifier, MultinomialNB, SVC, KNeighborsClassifier modellerinde test etmemizden oluşuyor. Verilerin vektörize yöntemleri gibi parametrelerini değiştirerek elde ettiğimiz sonuçları da kıyasladık. Verileri önce ön işlemden geçirdik ve bu ön işlemden sonra verileri rastgele karıştırdık ve sonrasında %80 eğitim ve %20 test verisi olarak böldük. Bunun sonucunda da belirli sayısal sonuçlar elde ettik. Elde ettiğimiz sonuçları da matplotlib yardımıyla çizdirdik.

Modellerin bizim veri setimiz için en başarılı parametrelerini sklearn.model_selection GridSearchCV yardımıyla bulduk. Örneğin KNN için "param_grid_KNN = {'n_neighbors': [3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41]}" parametrelerini denedik. Burada bize veri kümesinde en başarılı olunan parametreyi GridSearchCV döndürdü ve onunla yolumuza devam ettik.

KNN için ilk denememizde en başarılı parametre {'n_neighbors': 29}, SVM için {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}, NB için {'alpha': 0.1}, DT için {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}, RF içinse {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100} olmuştur.

Özellik Seçimi nedir?

Özellik seçimi, veri setindeki özelliklerin sayısını azaltmak veya daha iyi bir özellik alt kümesi oluşturmak için kullanılan bir yöntemdir. Veri setlerindeki özellik sayısının azaltılması, aşırı öğrenmeyi önlemeye, eğitim süresini kısaltmaya, daha az bellek tüketimine ve daha az karmaşıklığa neden olabilir. Özellik seçimi, veri setinin boyutunu azaltarak, gereksiz veya tekrarlayan özelliklerin çıkarılmasını sağlar ve bu da daha iyi bir performansa yol açabilir. Özellik seçimi, aynı zamanda, önemli özellikleri vurgulamak için de kullanılabilir. Bu, özelliklerin yalnızca bir alt kümesinin kullanıldığı durumlarda bile, yüksek doğruluklu modeller oluşturulabileceği anlamına gelir. Biz projemizde SelectKBest özellik seçim yöntemini kullandık.

SelectKBest, verilerin özelliklerini (features) seçmek için kullanılan bir özellik seçimi yöntemidir. Verilerin özelliklerini (features) seçerken istatistiksel yöntemler kullanarak, en önemli özelliklerin seçilmesine olanak sağlar. SelectKBest, veri setindeki her özelliğin, hedef değişkenle (target variable) olan ilişkisini hesaplar ve en yüksek skorlara sahip k özelliği seçer. SelectKBest sınıfı, özellik seçimi için iki farklı yöntem sağlar: chi-kare istatistiğini kullanarak sınıflandırma özelliklerini seçmek veya ANOVA f istatistiğini kullanarak regresyon özelliklerini seçmek.

Chi-kare skorlama fonksiyonu, kategorik özellikler arasındaki bağımlılıkları ölçer ve her bir öznitelik için bir p-değer hesaplar. Seçilen öznitelikler, sınıflandırma probleminde daha yüksek bir ayrım gücüne sahip olan ve sınıflar arasındaki ilişkiyi daha iyi yansıtan özelliklerdir. Ancak, seçilen özniteliklerin yanlış seçilmesi durumunda, sınıflandırma performansını olumsuz etkileyebilir.

3 Veri Seti

Veri setini amazonun aşağıdaki linkli ürünlerinin arasından rastgele şekilde seçerek oluşturduk.

https://www.amazon.com/Gildan-T-Shirts-Multipack-Style-G1100/product-reviews/B0BZDLXKTS/ref=cm_cr_getr_d_paging_btm_next_4?ie=UTF8&reviewerType=all_reviews&pageNumber=4&filterByStar=one_star

https://www.amazon.com/Amazon-Essentials-Slim-Fit-Short-Sleeve-Crewneck/dp/B07757W7CC/ref=sr_1_1_sspa?crid=3KZGZK092NTVI&keywords=t%2Bshirt&qid=1683

200428&sprefix=tshir%2Caps%2C644&sr=8-1-spons&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUE0QzBGQ1FWNlZDUDEmZW5jcnlwdGVkSWQ9QTA3MDcwNzIyWTBDMUJCVVZMTThHJmVuY3J5cHRlZEFkSWQ9QTA1NTgyOTdPWTBUNllXN1RUVDMmd2lkZ2V0TmFtZT1zcF9hdGYmYWN0aW9uPWNsaWNrUmVkaXJlY3QmZG9Ob3RMb2dDbGljaz10cnVl&th=1&psc=1

Veri setinde {1: 35, 2: 32, 3: 28, 4: 35, 5: 70} sayılarında labellar var. Bu labellar ürünün yorumuna verilen yıldızları temsil ediyor. Veri setinde bir de "comment" var. Bu da yorumu temsil ediyor. Aslında veri seti daha da geliştirilirse yapılan yorum için yıldız önerisi yapılmasına yönelik bir uygulama geliştirilirken kullanılabilir. Oluştuduğumuz veri setinden birkaç tane örnek aşağıdaki görselde verilmiştir.


```
    {
        "id":2,
        "comment":"I don't shop for white shirt's for myself and I usually stick to the basic brands like Hanes when buying them for my son. I haven't heard of this brand before but couldn't pass up the deal for 6 shirts. The M fit him perfectly, they feel nice and he was happy with the fit. He prefers these over his other ones now. I wanted to wash them first before the review to see if they shrank or stayed true to size, they stayed true. Very happy with this purchase.",
        "star":5
    },
    {
        "id":3,
        "comment":"I wear black t-shirts exclusively under my polos everyday. It's just a look that I have grown to like. I also like it when my t-shirt collars keep their shape. The Gildan brand does just that. This is what I was looking for in an under shirt and they delivered. The price is right too. I usually wear a large and that's what I ordered. The fit was just right. I would recommend this product. Have a great day!",
        "star":5
    },
```


Veri Setinden Örneklem

4 Sonuçlar

Programı ilk çalıştırdığımızda elde ettiğimiz çıktı şu şekilde:

Training set size: 160

Test set size: 40

All label Counts: {2: 32, 1: 35, 4: 35, 3: 28, 5: 70}

Train label Counts: {1: 28, 4: 28, 5: 56, 2: 26, 3: 22}

Test label Counts: {1: 7, 4: 7, 5: 14, 3: 6, 2: 6}

Fitting 5 folds for each of 20 candidates, totalling 100 fits

{'n_neighbors': 29}

Best parameters for SVM: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}

Fitting 10 folds for each of 20 candidates, totalling 200 fits

Best parameters for NB: {'alpha': 0.1}

Best parameters for DT: {'criterion': 'entropy', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}

Best parameters for RF: {'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}

==========

FOR KNN MODEL

Real accuraccy is -> 0.425

Cross val score is-> [0.3 0.45 0.4 0.3 0.4 0.5 0.45 0.45 0.3 0.35]

==

knn model saved...

==

FOR SVM MODEL

Real accuraccy is -> 0.525

Cross val score is-> [0.55 0.45 0.55 0.35 0.35 0.55 0.55 0.6 0.3 0.55]

==

svm model saved...

==

FOR NB MODEL

Real accuraccy is -> 0.575

Cross val score is-> [0.4 0.45 0.45 0.4 0.6 0.6 0.65 0.55 0.3 0.6]

==

nb model saved...

==

FOR DT MODEL

Real accuraccy is -> 0.325

Cross val score is-> [0.45 0.45 0.25 0.35 0.5 0.35 0.35 0.35 0.45 0.15]

==

dt model saved...

==

FOR RF MODEL

Real accuraccy is -> 0.5

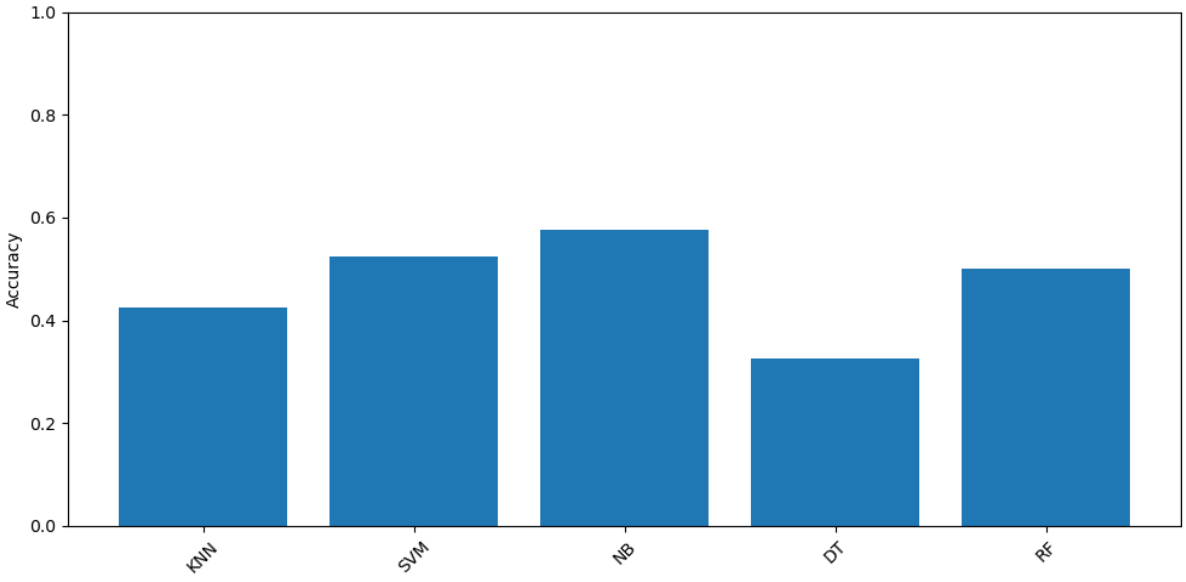
Cross val score is-> [0.4 0.4 0.4 0.5 0.35 0.35 0.4 0.4 0.35 0.45]

==

rf model saved...

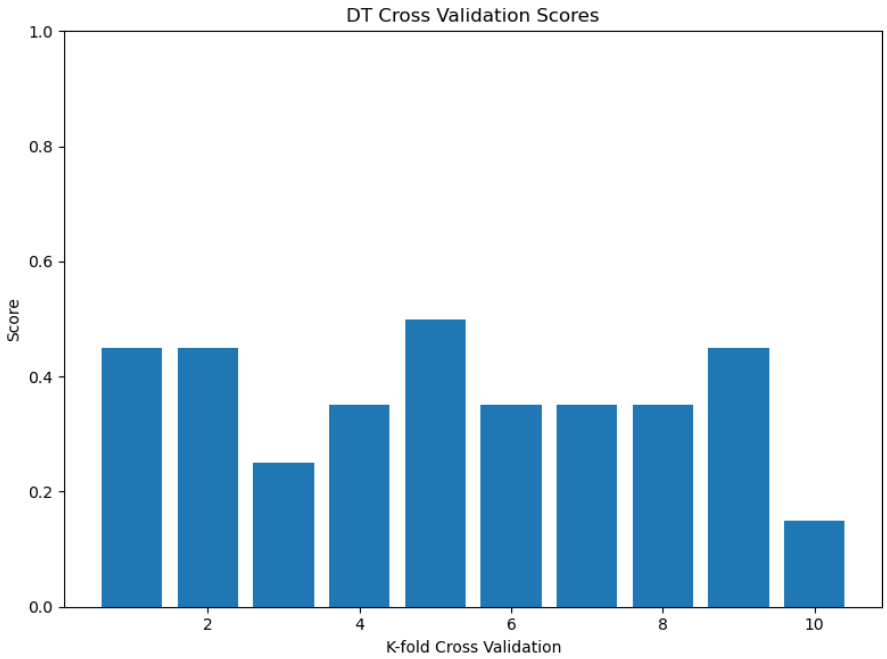
Vectorizer saved...

Bununla beraber her model için çizdirilmiş başarım sonuçları da aşağıdaki gibi:

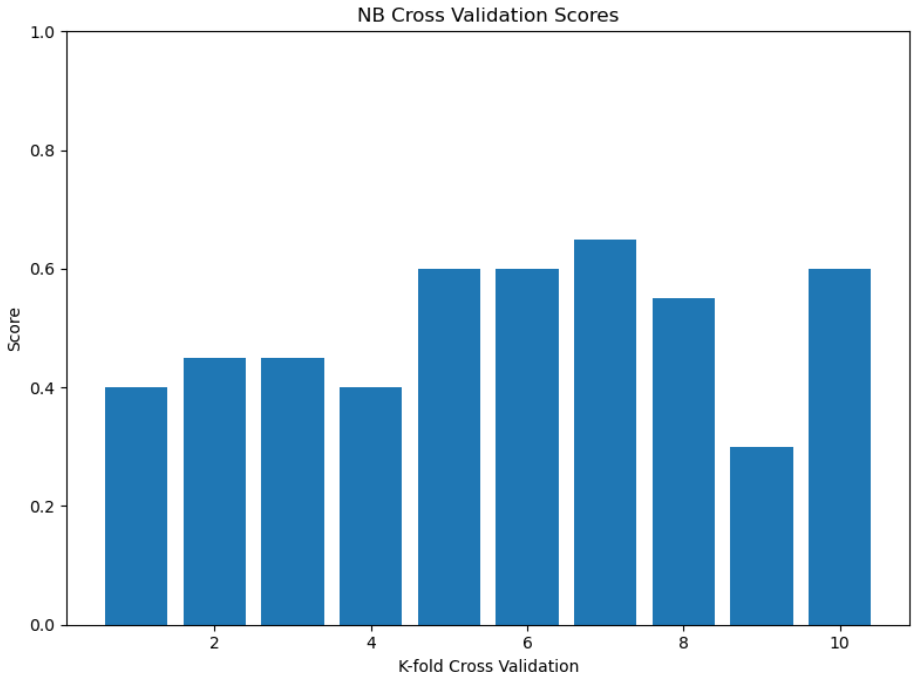


Tüm modeller için başarım skorları

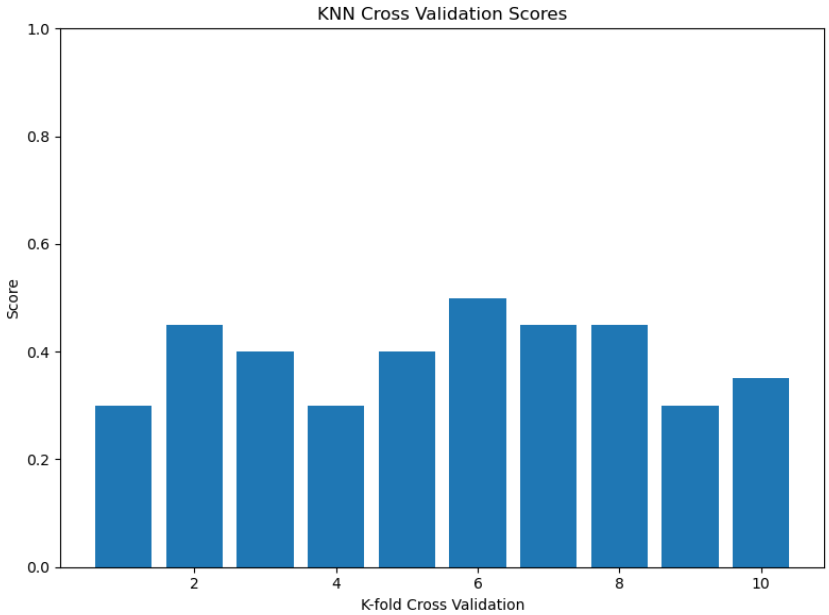
Burada görüldüğü üzre en başarılı model Multinomial Naive Bayes olmuştur. En başarısız model ise Decision Tree olmuştur. İlk sonuçlara göre veri kümemizle beraber Multinomial Naive Bayes modeli kullanmak çok mantıklı olacaktır. Bu sonuçları **TfidfVectorizer** kullanarak aldık. Aynı zamanda bu verilere ANOVA testi uyguladığımızda F-statistic: 4.008, p-value: 0.007 çıktılarını elde ettiğimiz için, örneklem grupları arasında anlamlı bir fark olduğu sonucuna varılabilir.



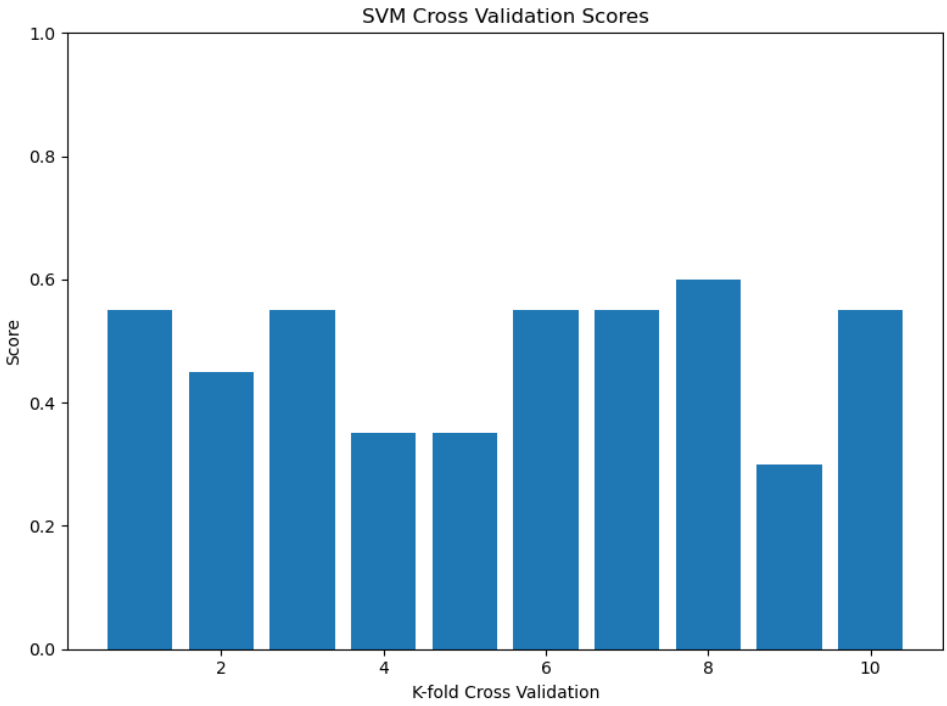
Score değerleri genel olarak yakın ve tutarlı sonuçlar alındığı görülmüştür. Ancak katman 10 değerinde iken score değerinin ortalamanın oldukça altında olduğu görülmüştür. Bu katman incelendikten sonra özellik seçimi gözden geçirilerek veya ağaç derinliği düğüm bölme kriterleri gibi parametreler güncellenerek daha tutarlı sonuç elde edilmesi sağlanabilir.



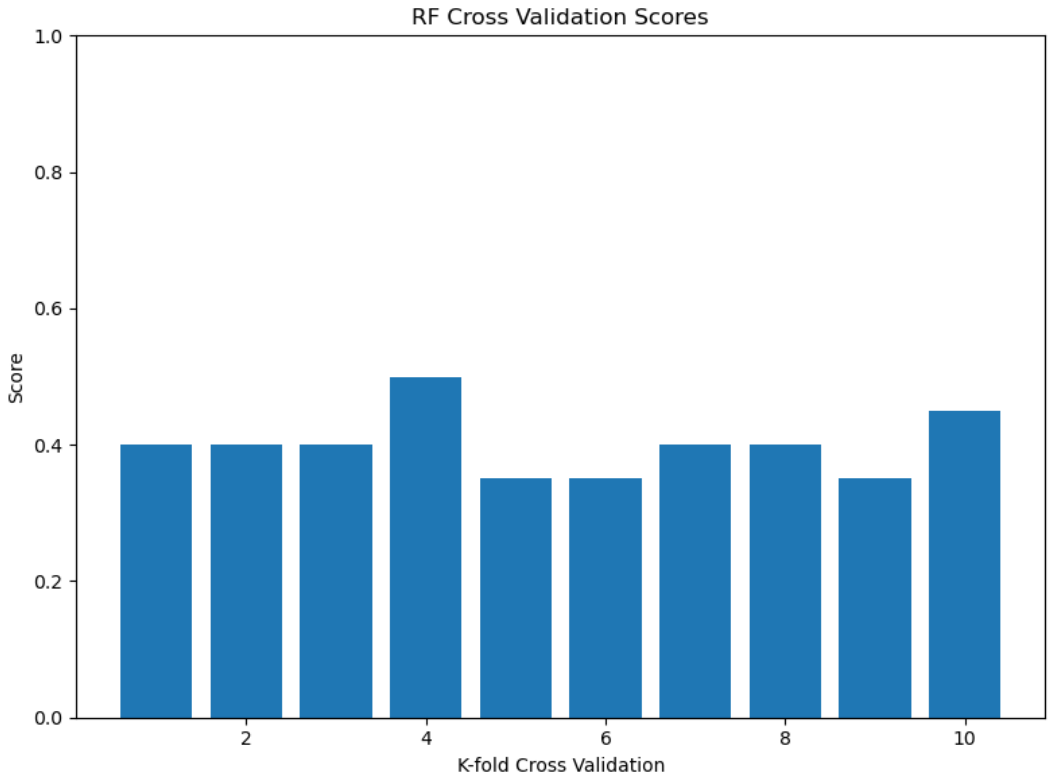
Naive Bayes karar ağaçlarına göre genel olarak daha yüksek bir score değeri yakaladığını ve score değerlerinin birbirine yakın olduğunu gözlemledik. Özellik seçimleri gözden geçirilerek daha iyi sonuçlara ulaşılabileceğini gördük.



KNN için grafik incelendiğinde her katman için score değerlerinin yakın çıktığı görülmektedir, böylece modelin tutarlı olduğu söylenebilir. K hiper parametresi güncellenerek ortalamada daha yüksek bir score değeri elde edilebilir.



Score değerleri incelendiğinde bazı katmanlarda değerlerin daha düşük olduğu görülmüştür. Bu farklılıklar veri dağılımında bu katmanlarda örnek sayısının az olmasından dolayı oluşabilir. Hiperparametreler gözden geçirilerek score değerlerinin yükselmesi sağlanabilir.



Grafik incelendiğinde oldukça yakın score değerleri elde edilmiştir. Bu da veri setinin oldukça tutarlı olduğunu göstermektedir.

Sonrasında vektörize yöntemini değiştirerek bunların başarıma etkisini gözlemlemek istedik ve CountVectorizer kullanarak modelleri yeniden eğittik. Konsol çıktılarımız aşağıdaki şekilde oldu:
Training set size: 160

Test set size: 40

All label Counts: {2: 32, 5: 70, 1: 35, 4: 35, 3: 28}

Train label Counts: {1: 28, 4: 28, 5: 56, 2: 26, 3: 22}

Test label Counts: {1: 7, 4: 7, 5: 14, 3: 6, 2: 6}

Fitting 5 folds for each of 20 candidates, totalling 100 fits

{'n_neighbors': 3}

Best parameters for SVM: {'C': 1, 'gamma': 0.1, 'kernel': 'sigmoid'}

Fitting 10 folds for each of 20 candidates, totalling 200 fits

Best parameters for NB: {'alpha': 2}

Best parameters for DT: {'criterion': 'gini', 'max_depth': 15, 'min_samples_leaf': 4, 'min_samples_split': 10}

Best parameters for RF: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 50}

==========

FOR KNN MODEL

Real accuraccy is -> 0.325

Cross val score is-> [0.5 0.4 0.4 0.35 0.3 0.35 0.4 0.4 0.4 0.35]

==========

knn model saved...

==========

FOR SVM MODEL

Real accuraccy is -> 0.45

Cross val score is-> [0.45 0.5 0.55 0.35 0.45 0.35 0.4 0.3 0.35 0.6]

==========

svm model saved...

==========

FOR NB MODEL

Real accuraccy is -> 0.5

Cross val score is-> [0.55 0.55 0.65 0.45 0.35 0.45 0.45 0.4 0.45 0.65]

==========

nb model saved...

==========

FOR DT MODEL

Real accuraccy is -> 0.325

Cross val score is-> [0.25 0.15 0.4 0.4 0.25 0.2 0.4 0.15 0.3 0.3]

==========

dt model saved...

==========

FOR RF MODEL

Real accuraccy is -> 0.425

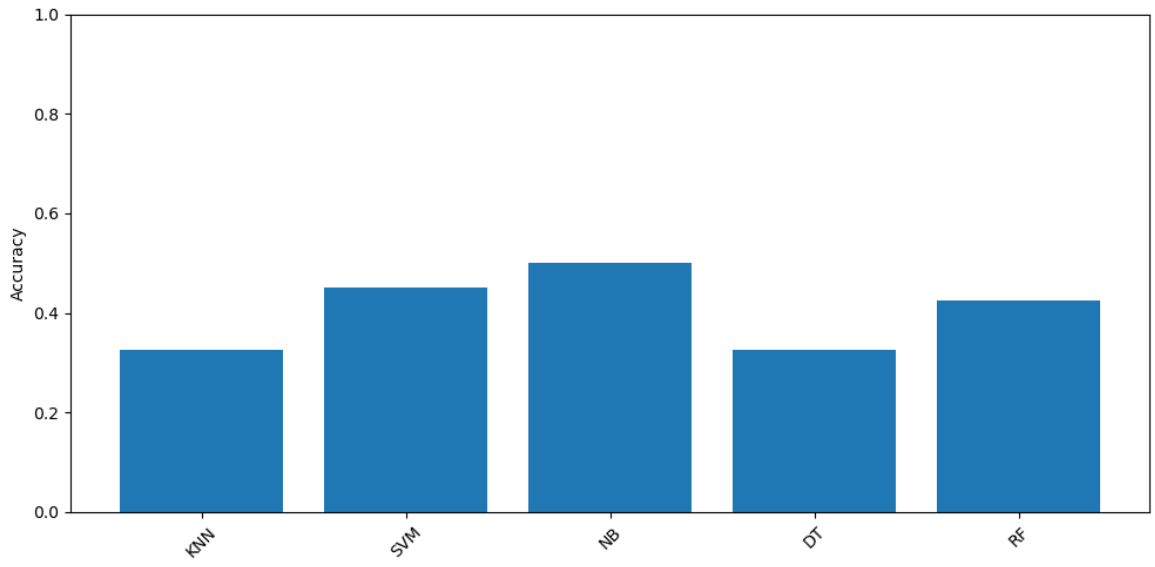
Cross val score is-> [0.55 0.4 0.4 0.35 0.4 0.3 0.4 0.4 0.4 0.45]

==========

rf model saved...

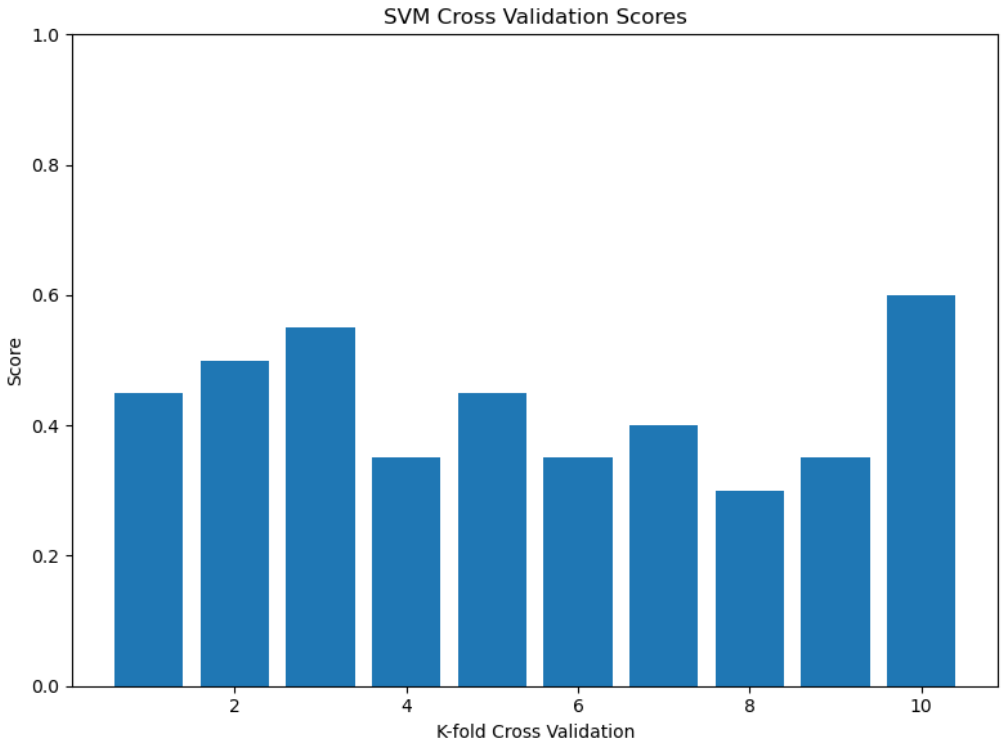
Vectorizer saved...

Grafik çıktılarına bakıldığındaysa:

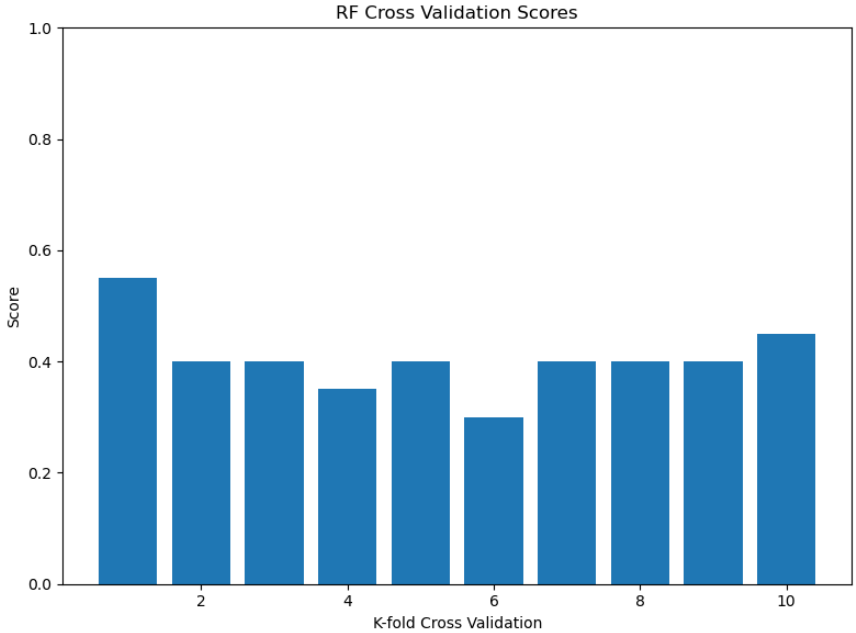


Tüm modeller için başarım skorları

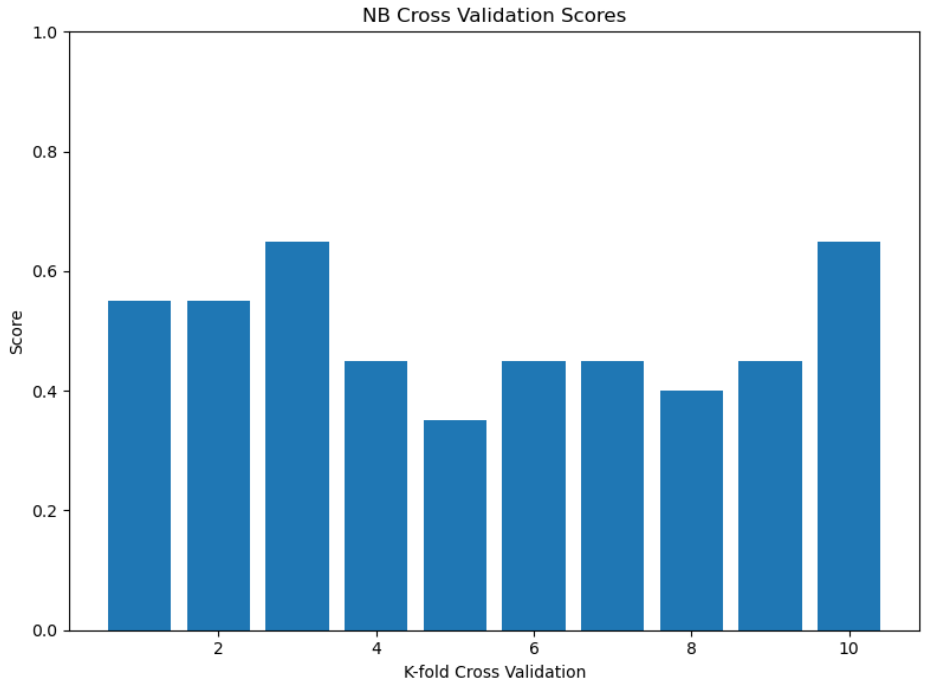
Burada görüldüğü üzere en başarılı model yine Multinomial Naive Bayes olmuştur. En başarısız model ise Decision Tree olmuştur. İlk sonuçlara göre veri kümemizle beraber Multinomial Naive Bayes modeli kullanmak çok mantıklı olacaktır. Bu sonuçları **CountVectorizer** kullanarak aldık. Grafiklerden de anlaşılacağı üzre, **TfidfVectorizer, CountVectorizer'a** göre çok daha başarılı sonuçlar elde etti. Aynı zamanda bu verilere ANOVA testi uyguladığımızda F-statistic: 38.003 ve p-value: 0.000 çıktılarını elde ettiğimiz için, örneklem grupları arasında anlamlı bir fark olduğu sonucuna varılabilir.



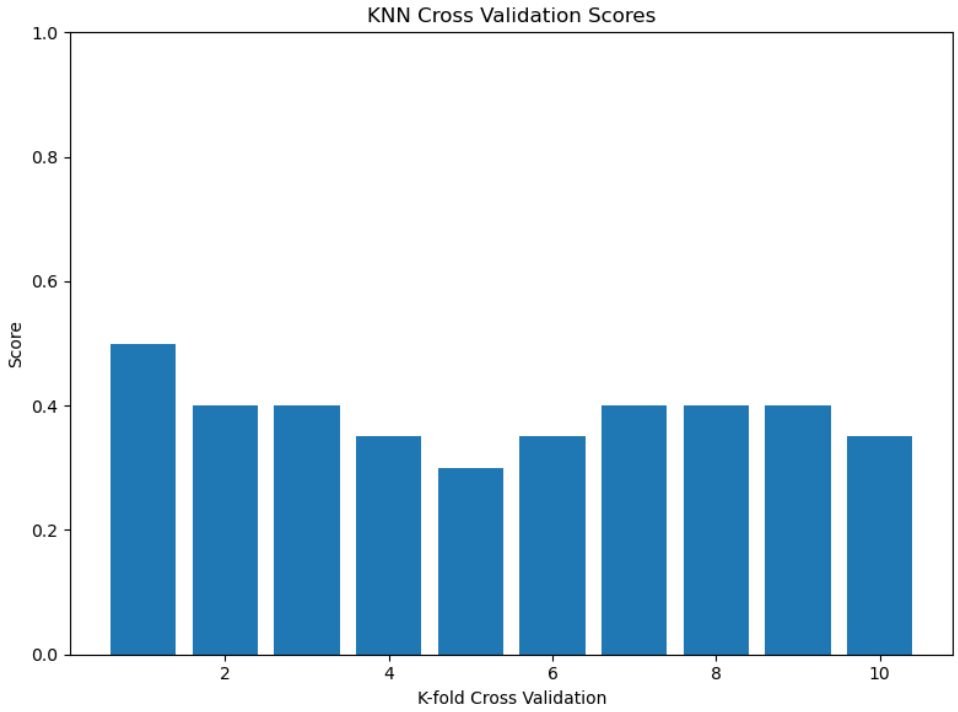
Score değerleri incelediğimizde vektörize yöntemini değiştirmemin bu model için katmanların dağılımı ve tutarlılığında önemli bir değişime sebep olmadığını gözlemledik.



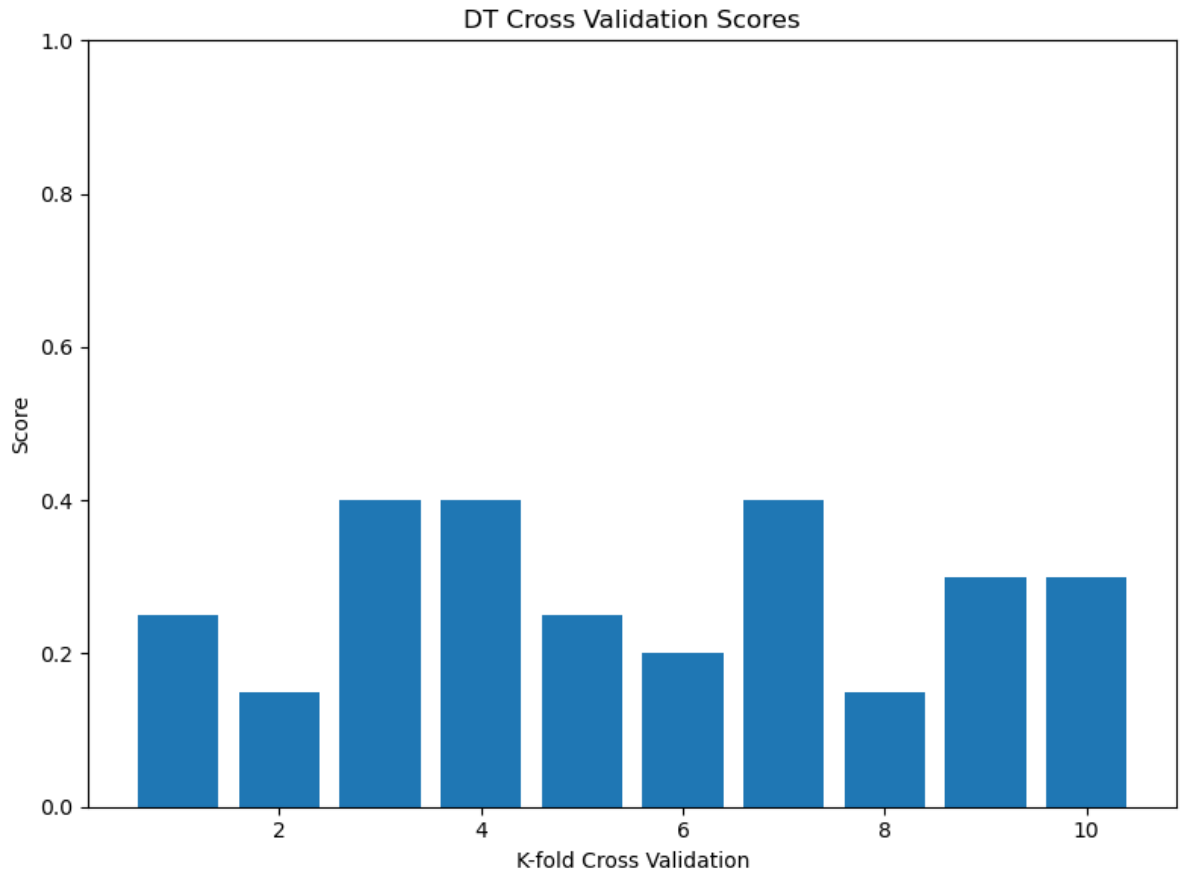
Score değerlerini incelediğimizde bu vektörize yönteminde bu model için katmanlar arası score tutarlılığının daha düşük olduğunu gözlemledik.



Score değerlerini incelediğimizde score değerlerinde vektörize yöntemi değiştiğinde düşük olan değerlerin genel olarak yükseldiğini düşük değerlerin ise yükseldiğini gözlemledik. Ancak katman değeri 10 olduğunda score değerinde kayda değer bir değişim olmadığını gözlemledik.



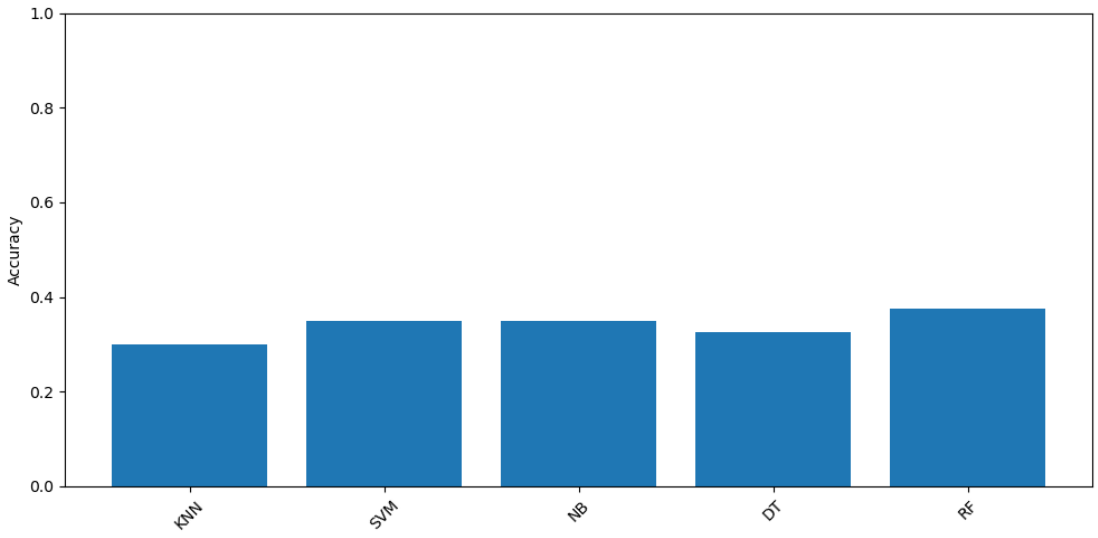
Grafiği incelediğimizde aynı şekilde vektörize yöntemi değiştiğinde yüksek değerlerin düştüğünü, düşük değerlerin ise yükseldiğini gözlemledik. Vektörize yönteminin değişmesinin her bir katmanda bulunan verilere bağlı olarak score değerlerini etkilediğini gözlemledik.



Değerleri incelediğimizde vektörize değerleri değiştiğinde score değerlerinin daha tutarsız şekilde çıktığını gözlemledik. Ayrıca ortalama score değerlerinin da düşmüş olduğunu gözlemledik.

Özellik seçimi:

Özellik seçminde SelectKBest(chi2, k=35) şeklinde bir seçim algoritması kullandık. Bununla beraber modellerin öğrenme süreci gözle görülür bir biçimde hızlandı. Ama bizim veri kümemiz için model başarımları oldukça düştü. Aşağıdaki resimde de görüldüğü üzre özellik seçimiden sonra, başarısı ne kadar önceki duruma göre düşse de, en başarılı model RF olmuş oldu. Ama burada modellerin başarımı birbirlerine oldukça yakın. SVM ve NB neredeyse aynı başarıma sahipken, en başarısız model bu sefer KNN oldu.



Tüm modeller için başarım skorları

Normalizasyon:

Şimdiye kadarki tüm sonuçları, verileri bir ön işlemden geçirmek suretiyle elde etmiştik. Bu ön işleme adımınını yapmadığımızda elde edilen konsol çıktısı aşağıdaki gibidir.

Training set size: 160

Test set size: 40

All label Counts: {2: 32, 1: 35, 4: 35, 3: 28, 5: 70}

Train label Counts: {1: 28, 4: 28, 5: 56, 2: 26, 3: 22}

Test label Counts: {1: 7, 4: 7, 5: 14, 3: 6, 2: 6}

Fitting 5 folds for each of 20 candidates, totalling 100 fits

{'n_neighbors': 41}

Best parameters for SVM: {'C': 10, 'gamma': 1, 'kernel': 'sigmoid'}

Fitting 10 folds for each of 20 candidates, totalling 200 fits

Best parameters for NB: {'alpha': 0.1}

Best parameters for DT: {'criterion': 'gini', 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 10}

Best parameters for RF: {'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}

==========

FOR KNN MODEL

Real accuraccy is -> 0.45

Cross val score is-> [0.5 0.35 0.55 0.45 0.3 0.45 0.45 0.45 0.45 0.45]

==

knn model saved...

==

FOR SVM MODEL

Real accuraccy is -> 0.475

Cross val score is-> [0.35 0.5 0.5 0.65 0.4 0.55 0.5 0.55 0.45 0.6]

==

svm model saved...

==

FOR NB MODEL

Real accuraccy is -> 0.425

Cross val score is-> [0.5 0.45 0.45 0.7 0.45 0.5 0.55 0.4 0.4 0.35]

==

nb model saved...

==

FOR DT MODEL

Real accuraccy is -> 0.475

Cross val score is-> [0.25 0.4 0.5 0.5 0.35 0.5 0.45 0.35 0.45 0.5]

==

dt model saved...

==

FOR RF MODEL

Real accuraccy is -> 0.375

Cross val score is-> [0.4 0.4 0.55 0.55 0.45 0.45 0.55 0.4 0.45 0.35]

==

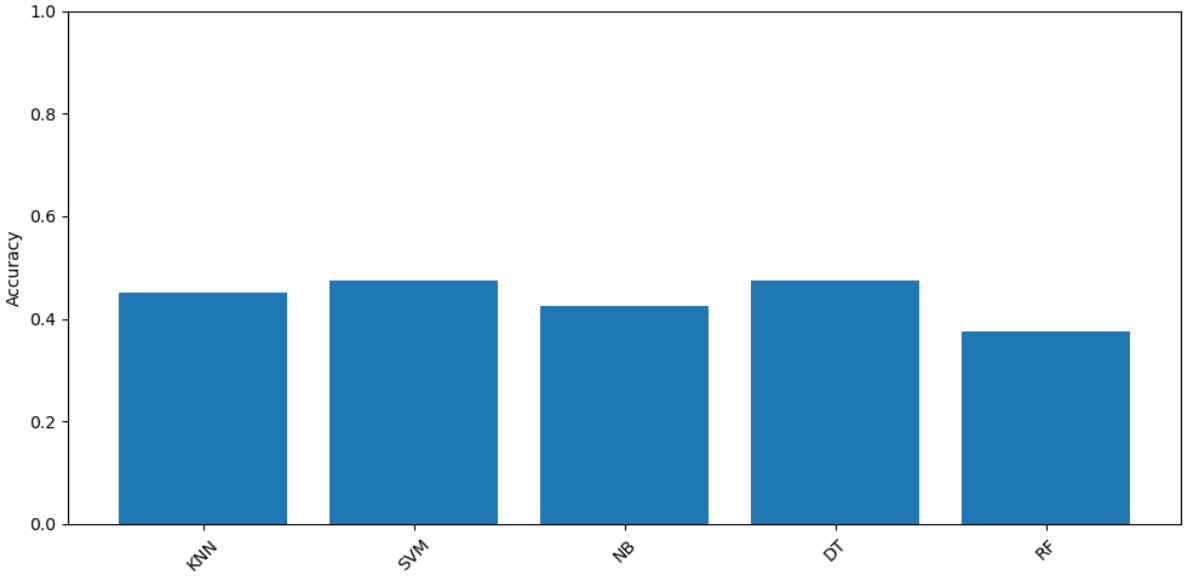
rf model saved...

Vectorizer saved...

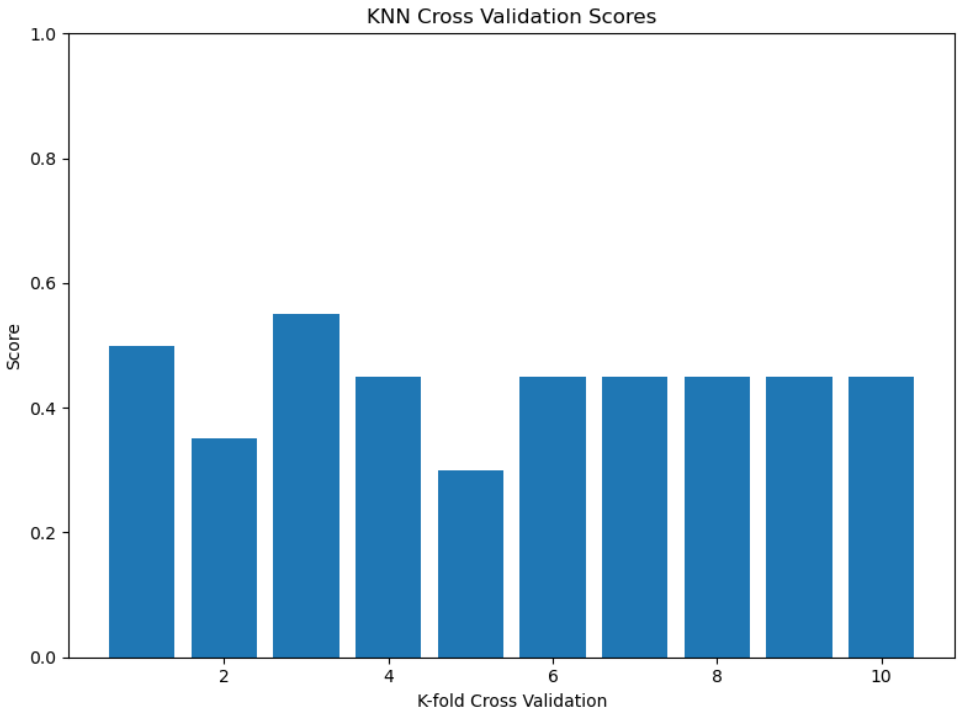
F-statistic: 1.388

p-value: 0.253

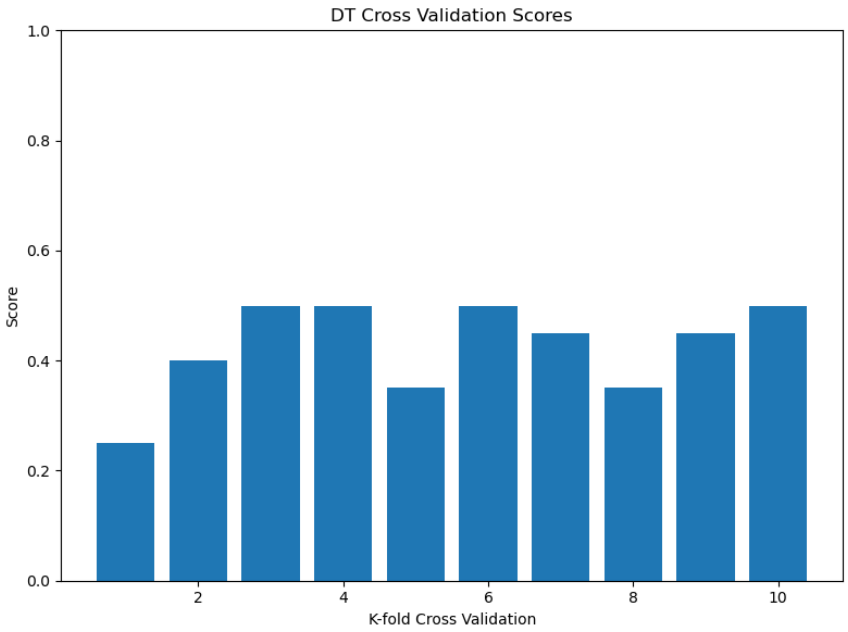
Görsel sonuçlara baktığımızdaysa başarım aşağıdaki resimde de görüldüğü üzre bir miktar düşmektedir. Özellikle metin verilerinde, verileri ön işlemden geçirmek başarımı oldukça artırmaktadır. Ama bizim veri kümemizdeki kelime sayısı belli bir seviyeden az olduğu için, bu sonuçlara çok büyük bir etki etmemiştir. Hatta Karar Ağacı için başarım yükselmiştir bile. Ama Destek Vektöre Makineleri, Naive Bayes ve Random Forest modelleri için başarımın belli ölçüde düştüğü gözlemlenmektedir.



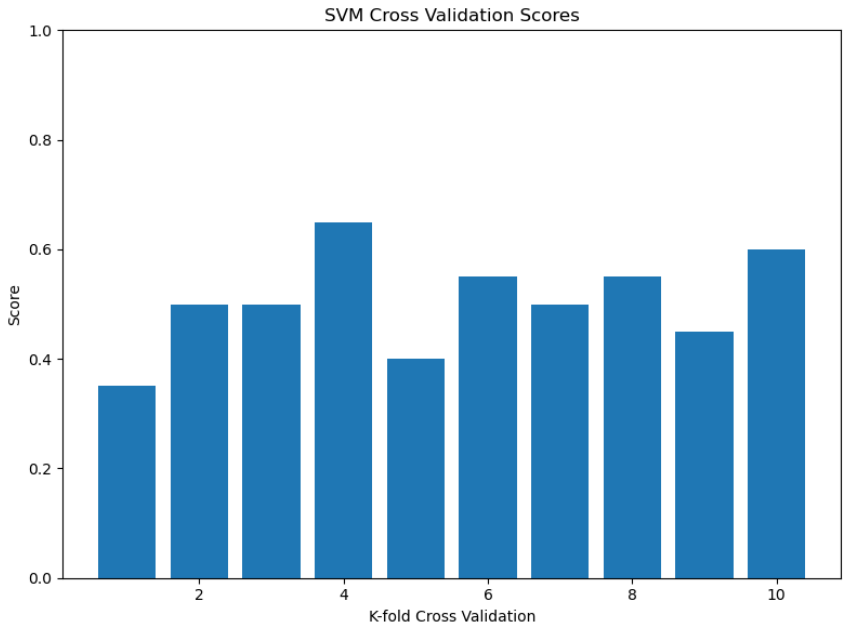
Tüm modeller için başarım skorları



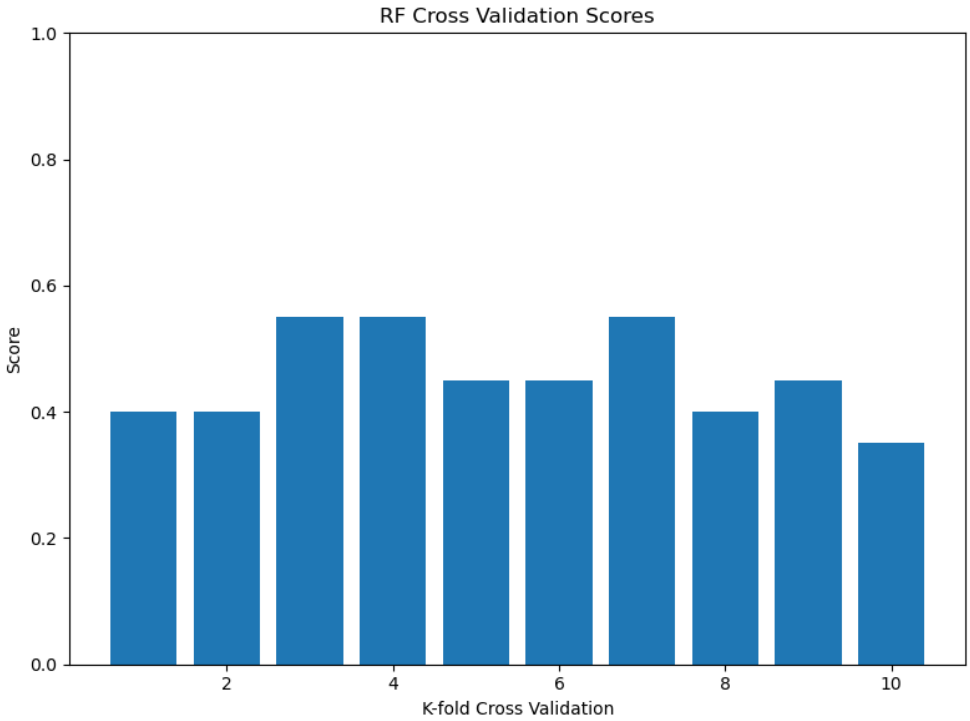
Score değerlerini incelediğimizde genel olarak önceki değerlere yakın değerler çıktığını gözlemledik. Ön işleme girmemesi durumunda katmanların daha tutarlı scrore değerlerinin çıktığını gözlemledik.



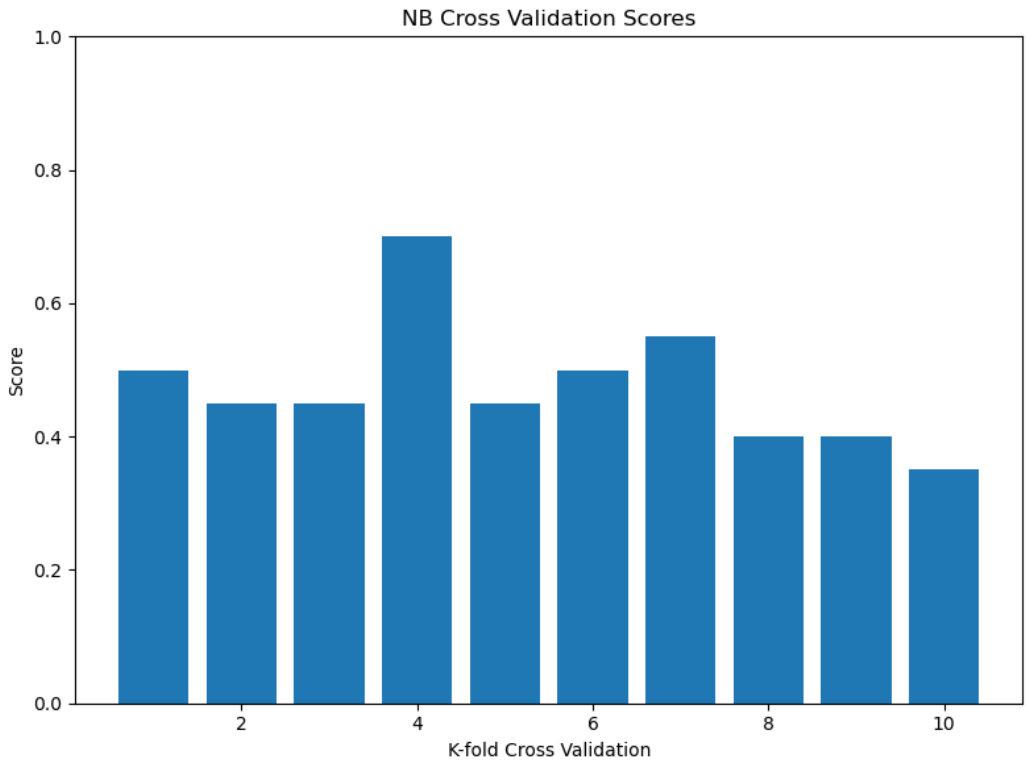
Score değerlerini incelediğimizde genel olarak önceki değerlere göre daha düşük değerler çıktığını gözlemledik. Ön işleme girmemesi durumunda katmanların score değerlerinin daha tutarsız çıktığını gözlemledik.



Score değerlerini incelediğimizde genel olarak önceki değerlere göre ortalamada yaklaşık değerler çıktığını gözlemledik. Ön işleme girmemesi durumunda katmanların score değerlerinin daha tutarsız çıktığını gözlemledik.



Score değerlerini incelediğimizde genel olarak önceki değerlere oldukça yakın değerler çıktığını gözlemledik. Ön işleme girmemesi durumunda katmanların score değerlerinin, ön işleme girdiğinde oluşan score değerleriyle neredeyse aynı tutarlılığa sahip değerlere sahip olduğunu gözlemledik.



Score değerlerini incelediğimizde ortalama score değerinin bir miktar düştüğünü gözlemledik. Ön işleme girmemesi durumunda katmanların score değerlerinin tutarsızlığının da yükseldiğini gözlemledik.

Son olarak genel başarımları yöntemlere göre bir tabloda toplayacak olursak:

	KNN	**SVM**	**NB**	**DT**	**RF**
TF-IDF Başarım(%)	42,5	52,5	57,5	32,5	50
COUNTER-V Başarım(%)	32,5	45	50	32,5	42,5
Özellik Seçimi Başarım(%)	30	35	35	32,5	37,5
Ön İşlemsiz Başarım(%)	45	47,5	42,5	47,5	37,5